Properties of Solution Concepts in Cooperative Concurrent Games

Szymon Kowara

Magdalen College
University of Oxford

Project Supervisor:
Dr. Julian Gutierrez

A thesis submitted for the degree of
Master of Computer Science

Trinity term, 2019

Acknowledgements

I would like to thank my supervisor, Dr. Julian Gutierrez, for suggesting this project on game theory, directing and guiding me throughout, and finally for reading and reviewing this thesis.

I would also like to thank my parents for funding my studies and supporting me through my whole degree.

I also want to thank my home city of Gdańsk for financially supporting my education at Oxford with the G.D. Fahrenheit Scholarship.

Abstract

Concurrent systems provide a framework for analysing numerous processes in nature, economics, or computing. Often times, the behaviour of agents can be explained with game-theoretical concepts. Well established solutions such as the Nash equilibrium play a big role in rational verification of such systems.

In this thesis, we are interested in model theoretic properties of cooperative solution concepts based on the core. We analyse its variation newly introduced to concurrent games - referred to as core$^+$. We prove a number of results about the core$^+$ regarding its existence and invariance under bisimilarity - a desirable property i.a. in model checking.

Lastly, we reflect on our results and suggest other cooperative solution concepts that could be considered in a similar context.

Contents

1 Introduction **3**

1.1 Context 3

1.2 Solution Concepts 4

1.2.1 Nash equilibrium 4

1.2.2 The core 4

1.2.3 Core$^+$ 5

1.3 Results 5

1.4 Structure of the Thesis 6

2 Preliminaries **7**

2.1 Concurrent Game Structures 7

2.2 Bisimilarity 8

2.3 Computations, Runs, and Traces 9

2.4 Strategies and Goals 10

2.5 Core$^+$ 12

2.6 Linear Temporal Logic 13

3 Non-Emptiness Problem **15**

3.1 Deviation Graphs 15

3.2 Three-player Case 17

3.3 General case 26

3.3.1 Deviation analysis 26
3.3.2 Five-player case 29
3.3.3 Four-player case 33

4 Bisimulation Invariance Problem **37**

4.1 Computation and Trace Based Strategies 37
4.2 Computation Based Goals 38
4.3 Run and Trace Based Goals 41

5 Non-Emptiness Invariance Under Bisimilarity **45**

5.1 Computation and Trace Based Strategies 45
5.2 Run Based Strategies 46

6 Other Cooperative Solution Concepts **48**

6.1 Core* 49
6.2 Core' 52

7 Conclusions **55**

7.1 Summary of Results 55
7.2 Open Questions 57
7.3 Different Solution Concepts 58
7.4 Related Work 59

Chapter 1

Introduction

1.1 Context

Concurrent games are a well established framework for analysis and verification of multi-agent systems [14, 15, 25]. They represent an environment or a system that is continuously influenced by simultaneous decisions of agents who are attempting to achieve their goals [1, 26].

In rational verification of concurrent multi-agent systems, we usually ask which logical properties should hold under the assumption that players behave rationally. This is in turn decided by game-theoretical concepts [20, 23], whether in a cooperative or non-cooperative setting [2, 7, 12, 16].

Bisimilarity also plays a big role in verification and formal analysis of concurrent systems. From a logical perspective, bisimilar systems are considered behaviourally equivalent, as they satisfy the same temporal properties in respect to languages such as Linear Temporal Logic (LTL) or Computation Tree Logic (CTL) [17, 21]. It is hence reasonable to ask whether analysed equilibria of the game would also be invariant between bisimilar game structures.

Therefore, we study properties of game-theoretical solution concepts applied to concurrent game structures. We base on earlier work related to Nash equilibrium [12, 13] and α-core [16] and analyse the behaviour of a new concept, core$^+$.

1.2 Solution Concepts

1.2.1 Nash equilibrium

Nash equilibrium (NE) is probably the best established and most accepted solution concept in multiplayer games. It captures the rationality of play by assuming that agents act independently, in a non-cooperative setting, and are unable to make binding agreements or declarations. Indeed, the strategies of players are defined to be in equilibrium if no single player can deviate (change his strategy) and increase his payout.

The idea for this paper originates from earlier work [12] where Nash equilibrium in concurrent game structures was analysed in detail. It was shown that, in general, it does not satisfy the desirable bisimulation-invariace property, at least not for the most commonly used model of strategies. Alternative models were suggested to combat that problem.

1.2.2 The core

The reasearch was then extended to cooperative multi-player games [16]. The core, perhaps the most popular cooperative solution concept captures the idea of binding agreements by claiming that strategies should be in equilibrium (form a core-stable outcome) if no set of players can form a deviating coalition and profit.

Compared to conventional cooperative games [4], however, the core does not extend naturally to concurrent games, as here the payout of the coalition also depends on how other players react. Based on the concept of α-core [24], the outcome is defined to be core-stable if every potential deviating coalition can be punished by the remaining players.

Defined this way, the core turns out to provide more promising results than Nash equilibrium. The set of core-stable outcomes is firstly non-empty for every concurrent game structure, and secondly, it satisfies the same LTL properties between any two bisimilar games [16].

1.2.3 Core$^+$

Despite positive model theoretic properties, α-core faces significant criticism as a solution concept. In a world where players are expected to behave rationally, its lack of restrictions on punishments is often considered too weak. It is doubtful that a coalition would be formed if participants were required to act against their own interests when punishing deviators.

Motivated by this issue, a variation of the core that attempts to make punishment strategies more credible is proposed. Referred to as core$^+$, it accepts a punishment for a deviation only if the remaining (non-deviating) players are no worse off in the resulting outcome than they initially were [16]. This paper follows up on that suggestion by studying the model theoretic properties of core$^+$.

1.3 Results

Similarly to earlier study on Nash equilibrium [12], we analyse the properties of core$^+$ in respect to different models of strategies and goals. We show that, in general the set of core$^+$-stable outcomes can be empty, even though surprisingly, it is always non-empty for 3-player games. We find out that for the most common (run-based) model of strategies, the core$^+$ fails to preserve its LTL properties under bisimulation (just like NE). We reflect on the preservation of existence of any core$^+$-stable outcome between bisimilar games, noting the difference to the case of Nash equilibrium.

Lastly, we suggest two new solution concepts - core* and core' as alternatives to core$^+$, and show that the set of core*-stable outcomes is always non-empty.

1.4 Structure of the Thesis

This thesis is divided into 7 chapters. In the first chapter we give an introduction to the studied problems, explaining the motivation and the origin of the ideas. We briefly summarise the results and outline the structure of the rest of the paper.

In the second chapter we provide the theoretical background necessary for the study. We formally introduce concepts of concurrent games, bisimilarity, and a game-theoretic solution concept called core$^+$ - our main object of interest.

In Chapter 3 we present the main results of our work - answers to the model theoretic question of non-emptiness of core$^+$.

In Chapter 4 we follow that up with the results on invariance of the answer to the E-core$^+$ problem between bisimilar game structures - an important question in the context of rational verification.

In the fifth chapter we address the question of invariance of existence of a core$^+$-stable outcome between bisimilar games as it turns out to be significantly more interesting than in the case of Nash equilibria.

Finally, in Chapter 6 we talk about issues with core$^+$ in respect to some of its unintuitive properties and model theoretic results. We present our alternatives for cooperative solution concepts that could be considered in future studies.

In the concluding chapter we fully summarise the results, present remaining open questions for core$^+$, and give suggestions for future research. We also talk about past work related to the topic.

Chapter 2

Preliminaries

2.1 Concurrent Game Structures

Every *concurrent game structure* (CGS) consists of a set $Ag = \{1, ..., n\}$ of *agents* (*players*), a set St of *states* (with an *initial state* $s_0 \in St$), and a set AP of *propositional variables*. Moreover, for each player $i \in Ag$ and each state $s \in St$ we define a non-empty set of *actions* available to i (Ac_i) at state s (denoted $Ac_i(s)$). A *direction* is a tuple $d = (a_1, ..., a_n)$ with each $a_i \in Ac_i$ that denotes a choice of action by every player. The direction is *legal* at state $s \in St$ if for each i we have $a_i \in Ac_i(s)$.

The possible plays of the game are described by a *transition function* δ that for every state $s \in St$ and for every direction d legal at s uniquely determines a subsequent state $s' \in St$. Moreover, we define a *labelling function* λ that with every state associates a *valuation* $v \in 2^{AP}$.

We draw the CGS using a graph where nodes represent states, and are labeled with their valuations. Edges represent transitions, each edge labeled with the directions that cause the transition. For clarity, we will not draw self-loops in "final" states - every state without outgoing edges will be assumed to have a single action available for each player, and a corresponding direction leading to itself.

The example below illustrates a CGS with 2 players, 5 states, and 2 propositional variables. For instance, in the initial state - 1, both players can choose between

playing “a” and “b”. The state has valuation (0,0), and both players choosing “a” will form a direction (a,a) and result in a transition to state 3.

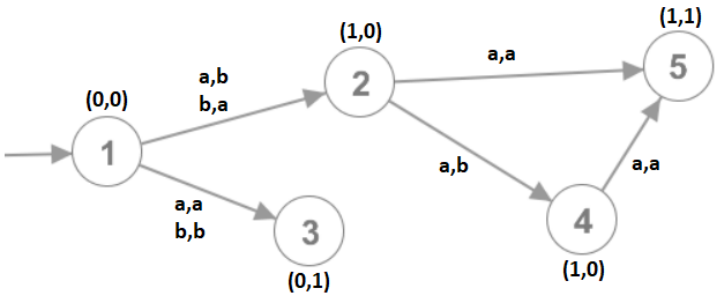

Figure 2.1: Example of a CGS with 2 players

2.2 Bisimilarity

Bisimilarity in concurrency is usually defined over Kripke structures or labelled transition systems [17, 21]. We make use of the extension of the definition to concurrent game structures [12].

Let $M = (Ag, St, s_0, AP, Ac, \delta, \lambda)$ and $M' = (Ag, St', s_0', AP, Ac', \delta', \lambda')$ be CGSs. A *bisimulation* between states $s \in St$ and $s' \in St'$ is a relation $R \subseteq St \times St'$ such that sRs' and the following hold for all $q, r \in St$ and $q', r' \in St'$:

- If qRq' then $\lambda(q) = \lambda'(q')$
- If qRq' and $\delta(q, d) = r$ for some direction d legal at q then d is also legal at q' and $\delta'(q', d) = p'$ for some $p' \in St'$ with rRp'
- If qRq' and $\delta(q', d) = r'$ for some direction d legal at q' then d is also legal at q and $\delta(q, d) = p$ for some $p \in St$ with pRr'

If there is a bisimulation between initial states s_0 and s_0' then we say that CGSs M

and M' are *bisimilar*, denoted $M \sim M'$. It is easy to check that this is an equivalence relation [2, 8, 21].

Figure 2.2 illustrates a very simple CGS with 1 player who gets to choose between "a" and "b" in states 1 and 2, but his choice "does not matter" in the context of bisimulation. Figure 2.3 then presents a bisimilar CGS.

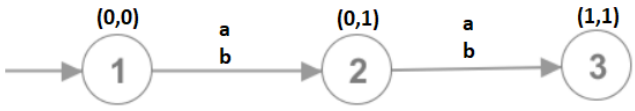

Figure 2.2: A simple CGS with 1 player

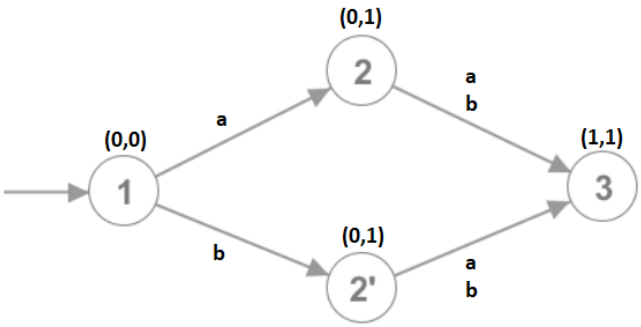

Figure 2.3: A CGS with 1 player bisimilar to Figure 2.2

2.3 Computations, Runs, and Traces

We differentiate between three levels of abstraction on which the play of a concurrent game can be tracked [3, 12]. Computations are the most informative, as they carry the exact decisions made by players at every step. Runs, the most common model in literature, carry the sequence of states visited during the game as a result of those decisions. Finally, traces only take note of the sequence of valuations at those states.

Formally, a *computation* κ on a CGS M is a (possibly infinite) sequence of directions $(d_1, d_2, ...)$, such that $\rho = (s_0, s_1, ...)$ is the corresponding sequence of states of M, with s_0 being the initial state, and for every $i \geq 0$, d_{i+1} is legal at s_i with $\delta(s_i, d_{i+1}) = s_{i+1}$. We use ϵ to denote the empty computation.

Since δ is a (deterministic) transition function, such a sequence of states is unique given a computation. We then say that ρ is the *run* induced by κ. Similarly, since λ is also a function, for every such run there is a unique sequence of valuations $\tau = (v_0, v_1, ...)$ such that for every $i \geq 0$, $\lambda(s_i) = v_i$. We call τ the *trace* induced by ρ (or by κ).

In the case of the CGS in Figure 2.1 an example computation would be a sequence $(a, b)(a, b)(a, a)(*, *)^\omega$. The induced run would then be 1245^ω. The induced trace would be a sequence $(0, 0)(1, 0)(1, 0)(1, 1)^\omega$.

It is important to note that sets of computations and traces of bisimilar game structures always coincide, while sets of runs might be different [12].

2.4 Strategies and Goals

To apply game-theoretical concepts to the study of concurrent games, we need to model the players' strategies, goals, and preferences. As described in the previous section, we can use one of the three abstractions.

Let M be a CGS, and let *comps*, *runs*, *traces* be sets of finite computations, runs, and traces on M, respectively. A *computation-based strategy* f_i for a player $i \in Ag$ is a function that to every $\kappa \in comps$ assigns some action $a \in Ac_i(s)$ where s is the last state in the run induced by κ. Intuitively, when deciding what to do in a state s, the player is able to look at the entire history of players' decisions from the start of the game.

The most commonly used model, a *run-based strategy* for a player i is a function that to every $\rho \in runs$ assigns an action $a \in Ac_i(s)$ where s is the last state in ρ. Here, the agent can only look at the history of states, without knowing the exact choices that led to those states.

Finally, a *trace-based strategy* is a function that to every $\tau \in traces$ assigns an action a such that $a \in Ac_i(s)$ holds for the last state s of every run that induces τ. The player can only look at the history of valuations when deciding his next move. It is important to note that defining a trace-based strategy for every player is not possible in every CGS [13], but we will not consider such cases here.

To determine the result of the game, we need to fix the strategies chosen by all players. A *strategy profile*, or an *outcome* of a game M with $Ag = \{1, ..., n\}$ is then a tuple $f = (f_1, ..., f_n)$ where f_i denotes the strategy of player i. Importantly, the strategies of all players are required to be of the same abstraction type (i.e. computation, run, or trace-based) resulting in a computation, run, or trace-based strategy profile.

Regardless of the type, every outcome identifies a unique infinite computation on M. To retrieve it, recursively compute the sequence of directions $(d_1, d_2, ...)$ where:

- $d_1 = (a_1, ..., a_n)$ where $a_i = f_i(\epsilon)$, $a_i = f_i(s_0)$, or $a_i = f_i(\lambda(s_0))$ in the cases of computations, runs, or traces, respectively
- $d_i = (a_1, ..., a_n)$ where $a_i = f_i(h)$ and h is either the computation $(d_1, ..., d_{i-1})$, or the run/trace induced by it

Write $\kappa(f)$, $\rho(f)$, $\tau(f)$ for the computation indentified by an outcome f, the run, and the trace induced by this computation, respectively. We can then also base preferences of players on either computations, runs, or traces.

Given a CGS M with the set *comps* of computations, for each player $i \in Ag$ define his *goal set* $\Gamma_i \subseteq comps$. We say that the player prefers the computations in Γ_i to the ones not in Γ_i, but is indifferent otherwise. The tuple $G = (M, \Gamma_1, ..., \Gamma_n)$ is then called a *concurrent game*.

We say that the goals of players of G are *run-based* if for every two computations κ, κ' that induce the same run ρ, we have $\kappa \in \Gamma_i \iff \kappa' \in \Gamma_i$ for each player i. We say that the goals are *trace-based* if the equvalence holds for every two computations that induce the same trace. See that trace-based goals are also run-based, and run-based goals are also computation-based, but not necessarily vice versa.

We abbreviate the concept of the goal set to goal, and say that an outcome f *satisfies* the goal Γ_i if $\kappa(f) \in \Gamma_i$. Moreover, every outcome f uniquely defines a partition of Ag into a set of players who get their goal satisfied (*winners*), and a set of those who do not (*losers*), denoted $W(f)$ and $L(f)$, respectively.

2.5 Core$^+$

With those formalities in place, we can now define game-theoretic concepts for concurrent games. By a *joint strategy* for a subset of agents $A \subseteq Ag$ we mean a tuple $f_A = (f_{i_1}, ..., f_{i_m})$ with $A = \{i_1, ..., i_m\}$, where each f_k is a strategy for player k. Note that a joint strategy with $A = Ag$ is an outcome. Given an outcome f and $B \subseteq Ag$, we write f_{-B} to mean a joint strategy for $Ag \setminus B$ that is obtained by removing strategies of players in B from f.

For formal definitions of the Nash equilibrium and the core (extension of α-core to concurrent games) we refer to [16]. The main criticism of the core presented in that paper is increbility of punishments for deviations from the stable outcome. The core requires a deviation for a subset of losers to be beneficial regardless of the reaction

of all other players. This motivates an alternative concept, core$^+$ [16].

Formally, we say that f'_C, a joint strategy for some $C \subseteq Ag$ is a *beneficial deviation* from an outcome f if all of the following hold:

- $C \subseteq L(f)$
- $C \subseteq W(f_{-C}, f'_C)$
- for every joint (punishment) strategy f'_{-C} for $Ag \setminus C$, we have $W(f) \subseteq W(f'_{-C}, f'_C) \Longrightarrow C \subseteq W(f'_{-C}, f'_C)$

Intuitively, we claim for a deviation to be beneficial if it cannot be (credibly) punished. First two conditions ensure that it is indeed a rational deviation: all the deviators improve on their payout - they go from being losers in the initial outcome to winners in the target outcome. The third condition ensures that any potential punishment for the deviation will be incredible: (by contraposition) if any of the deviators loses as a result of the punishment, at least one of the winners from the initial outcome also loses.

Finally, we say that an outcome f is *core$^+$-stable*, or *in the core$^+$* if there does not exist a beneficial deviation from f. The *core$^+$-set* of a game G, also denoted *core$^+$*(G) is the set of all such outcomes. Most of this paper is dedicated to studying model theoretic properties of this concept.

2.6 Linear Temporal Logic

Linear Temportal Logic (LTL) is one of the most important tools in formal verification and model checking [9, 14, 15, 19, 25]. It is able to express the properties of a system that are related to time, such as “until”, “eventually” or “never”. LTL

formulas are evaluated on ω-words, infinite sequences of propositional valuations. Therefore, in the case of concurrent game structures, LTL formulas will be used to describe properties of infinite traces.

In rational verification, we are often interested about what temporal logic properties are satisfied by stable outcomes. In the case of core$^+$, we can define one of the problems as follows [16]:

E-core$^+$

Given: Game G, LTL formula ϕ.

Question: Is there an outcome $f \in core^+(G)$ such that $\tau(f)$ satisfies ϕ?

The paper on concurrent games studies the complexity of this problem, and how it relates to bisimilarity in the case of α-core [16]. We will do the latter for core$^+$.

Chapter 3

Non-Emptiness Problem

A crucial property of a solution is its existence. Indeed, in the context of game-theoretical solution concepts, a reasonable question to ask is whether an instance of a stable outcome can be found in every game in the considered model. For concurrent games, we know that the answer is negative in the case of Nash equilibria [12], but positive in the case of α-core [16]. Moreover, the paper on cooperative games solves the problem for core$^+$ in 2-player games. In this chapter we answer the question in the general case:

Is $core^+(G) \neq \emptyset$ *for every concurrent game* G*?*

3.1 Deviation Graphs

To reason about core$^+$, it is useful to have a framework that handles deviations and punishments in an understandable manner. We introduce the concept of deviation graphs, which we use to express necessary properties of a given game. Intuitively, looking at a deviation graph of a game we will be able to tell something about the outcomes that it necessarily has, and about necessary available beneficial deviations between those outcomes.

Let G be a concurrent game. Let S be the set of joint strategies of G, and $O \subseteq S$ be the set of outcomes. Consider a graph (V, E) where V is the set of vertices, and

$E \subseteq V \times V$ is the set of edges. We say that (V, E) is a *deviation graph* of G if there exist functions $o : V \to O$ and $s : E \to S$ such that for every $(v_1, v_2) \in E$, $s((v_1, v_2))$ is a beneficial deviation from $o(v_1)$ to $o(v_2)$.

When drawing a deviation graph for particular instances of o and s, we will label every vertex v with $W(o(v))$ - the set of winners in the corresponding outcome. Moreover, we will label every edge e with $s(e)$ - the corresponding deviation.

As a concrete example, we will recreate the core$^+$ non-emptiness proof for the 2-player case [16] using a deviation graph to visualise it.

Theorem 3.1. *core$^+(G) \neq \emptyset$ for every 2-player game G*

Proof. For a contradiction, suppose that $G = (M, \Gamma_1, \Gamma_2)$ is a 2-player game with no outcome in core$^+$. First, note that no outcome can satisfy both goals Γ_1 and Γ_2, as it would trivially be in the core$^+$. Moreover, there must be an outcome that satisfies at least one of Γ_1 and Γ_2, because otherwise there would be no beneficial deviation from any outcome, and hence every outcome would be in the core$^+$.

Without loss of generality, we may assume that there is an outcome (f_1, f_2) satisfying exactly Γ_1. But then that outcome cannot be in core$^+$, and so it must admit a beneficial deviation f_2' for the only loser, player 2. The resulting outcome, (f_1, f_2') must then satisfy exactly Γ_2. Again, that outcome must have a beneficial deviation f_1' for player 1 to an outcome (f_1', f_2') that satisfies exactly Γ_1. Therefore, the following would be a deviation graph of G:

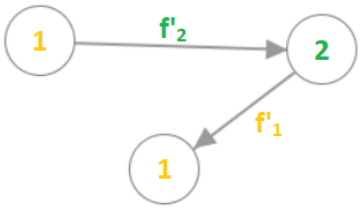

Figure 3.1: Deviation graph for a 2-player game with empty core$^+$

But now, we can see that only player 1 won in (f_1, f_2) and he also wins in (f_1', f_2'), hence f_1' is a credible punishment for f_2'. This contradicts an earlier assumption that f_2' was beneficial. □

3.2 Three-player Case

We are able to prove non-emptiness of core$^+$ in 2-player games using only the definition of core$^+$, without referring to the structure of the game. We will now generalise this proof to 3-player games by similarly assuming that there is a game without a core$^+$-stable outcome and then considering different cases. We will use deviation graphs as a high-level visualisation of the proof that accompanies the formal reasoning.

Theorem 3.2. *$core^+(G) \neq \emptyset$ for every 3-player game G*

Proof. For a contradiction, suppose that G is a 3-player game with no outcome in $core^+(G)$. Then every outcome (f_1, f_2, f_3) of G, where f_1, f_2, f_3 are strategies of 1, 2, and 3, respectively, must admit a beneficial deviation (from now on abbreviated to b.d.) for some subset of Ag. Let Γ_1, Γ_2 and Γ_3 be goals of 1, 2, and 3, respectively. We will now sequentially prove the following claims about G:

- *Claim 1:* G cannot have an outcome that satisfies all of Γ_1, Γ_2, Γ_3.
- *Claim 2:* G cannot only have outcomes that satisfy less than two of Γ_1, Γ_2, Γ_3.
- *Claim 3:* G cannot have an outcome that satisfies exactly two of Γ_1, Γ_2, Γ_3.

Proof of Claim 1:

If G had an outcome satisfying all of Γ_1, Γ_2, Γ_3, such an outcome would be in the

core$^+$, which is a contradiction. This claim will be used implicitly for the rest of the proof.

Proof of Claim 2:

Suppose, for contradiction, that G only has outcomes satisfying less than two of Γ_1, Γ_2, Γ_3. First, note that if G only had outcomes satisfying zero of Γ_1, Γ_2, Γ_3, none of those outcomes could admit a b.d., and so would all be in the core$^+$. Hence, G must have an outcome satisfying exactly one of Γ_1, Γ_2, Γ_3. Fix such an outcome as (f_1, f_2, f_3). Without loss of generality, we may assume that it satisfies only Γ_1.

Now, there must be a b.d. from this outcome. Since at least the deviators have to win in the resulting outcome, it must be an outcome satisfying exactly one of the goals. WLOG, we can assume that it is a deviation f_2' for 2 that satisfies only Γ_2.

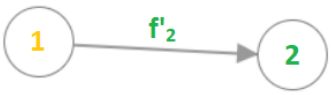

Figure 3.2: Deviation graph for G, Claim 2 (1)

The resulting outcome, (f_1, f_2', f_3) must again have a b.d. for 1 or 3 to an outcome satisfying only Γ_1, or only Γ_3, respectively. If former, however, then given such a deviation f_1', the joint strategy (f_1', f_3) for $\{1, 3\}$ would be a punishment for f_2'. Hence, we must have the latter, a b.d. f_3' for 3.

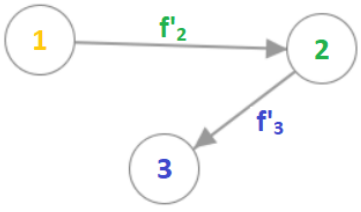

Figure 3.3: Deviation graph for G, Claim 2 (2)

The new outcome, (f_1, f_2', f_3') must once again admit a b.d. If it were to be a strategy

f_1' for 1 to an outcome satisfying Γ_1 only, then the joint strategy (f_1', f_3') would be a punishment for f_2'. Else, if the b.d. were a strategy f_2'' for 2 to an outcome satisfying Γ_2 only, then (f_1, f_2'') would constitute a punishment for f_3'. Either way, we reach a contradiction, which finishes the proof of *Claim 2*.

Proof of Claim 3:

We will first prove the following lemma about G:

- Lemma 3.1:

Let $\{a, b, c\} = \{1, 2, 3\}$ be the set Ag. There cannot be outcomes (f_a, f_b, f_c), (f_a, f_b, f_c'), (f_a', f_b, f_c') of G such that all of the following hold:

- (f_a, f_b, f_c) satisfies Γ_a and Γ_b but not Γ_c
- (f_a, f_b, f_c') satisfies Γ_c only
- (f_a', f_b, f_c') satisfies Γ_a only
- f_c' is a beneficial deviation for c from (f_a, f_b, f_c) to (f_a, f_b, f_c')
- f_a' is a b.d. for a from (f_a, f_b, f_c') to (f_a', f_b, f_c')

Note: The lemma is defined on a, b, c in order to be symmetric between players.

- Proof of Lemma 3.1:

For contradiction, assume otherwise. WLOG we can also assume that $a = 1$, $b = 2$, and $c = 3$. The negation of the lemma can then be summarised by a deviation graph.

Now, the third outcome, (f_1', f_2, f_3') must have a b.d. for 2, 3, or both. See that if it is a deviation (f_2', f_3'') (with possibly $f_2' = f_2$) to an outcome that satisfies Γ_3 but not Γ_1, then it would be a punishment for f_1'. Therefore, the only possible sets of winners in the new outcome are $\{2\}$, $\{1, 2\}$ and $\{1, 3\}$. Consider those cases as $3.1a$, $3.1b$, and $3.1c$, respectively.

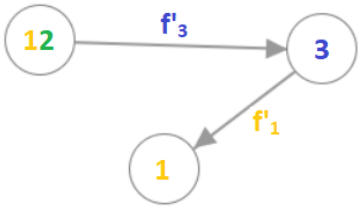

Figure 3.4: Deviation graph for G, Lemma 3.1

Case 3.1a

The only winner in the new outcome is 2. Hence, the b.d. must be f_2' by 2 only, to the outcome (f_1', f_2', f_3').

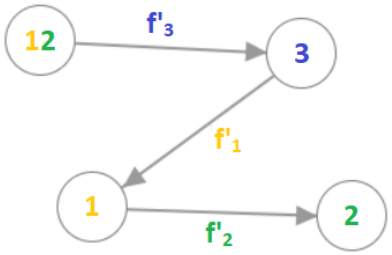

Figure 3.5: Deviation graph for G, Case 3.1a

Again, this outcome has to have a b.d. for 1, 3, or both. Let that deviation be (f_1'', f_3'') with possibly $f_1'' = f_1'$ or $f_3'' = f_3'$ but not both. Consider the following cases for the resulting outcome, (f_1'', f_2', f_3''):

- Only Γ_1, or exactly Γ_1 and Γ_3 are satisfied.

  Then (f_1'', f_3'') is a punishment for f_2', contradiction.

- Exactly Γ_1 and Γ_2 are satisfied.

  Here, since 3 loses, we must necessarily have $f_3'' = f_3'$ (the deviation is just by 1). In this case, the joint strategy (f_1'', f_2') can take us from (f_1, f_2, f_3') to (f_1'', f_2', f_3'') and form a punishment for f_3'. Contradiction.

- Only Γ_3, or exactly Γ_2 and Γ_3 are satisfied.

Since either way 1 loses, we must have $f_1'' = f_1'$ and the joint strategy (f_2', f_3'') can take us from (f_1', f_2, f_3') to (f_1'', f_2', f_3'') and form a punishment for f_1'. Contradiction.

Therefore, *Case 3.1a* leads to a contradiction.

Case 3.1b

The winners in the new outcome are 1 and 2. Since 1 was a winner in (f_1', f_2, f_3') already, the b.d. must once again be f_2' by 2 only, to the outcome (f_1', f_2', f_3').

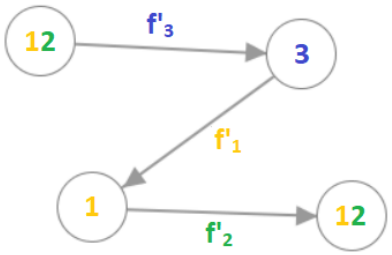

Figure 3.6: Deviation graph for G, Case 3.1b

But now, we can see that the joint strategy (f_1', f_2') is a punishment for f_3' (both 1 and 2 remain winners). Contradiction.

Case 3.1c

The winners in the new outcome are 1 and 3. Since 1 was a winner in (f_1', f_2, f_3') already, the b.d. must this time be f_3'' by 3 only, to the outcome (f_1', f_2, f_3'').

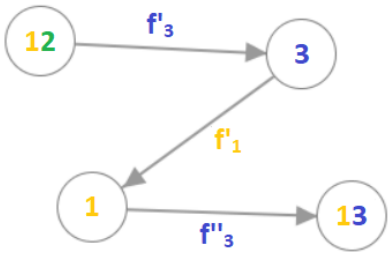

Figure 3.7: Deviation graph for G, Case 3.1c (1)

The only loser, 2, must have a b.d. f_2' from this outcome. Now, see that:

- If exactly Γ_1 and Γ_2 were satisfied in the resulting outcome - (f_1', f_2', f_3''), then the joint strategy (f_1', f_2') would be a punishment for f_3'.
- If exactly Γ_2 and Γ_3 were satisfied in (f_1', f_2', f_3'') then the joint strategy (f_2', f_3'') would be a punishment for f_1'.

Hence, 2 must be the only winner in (f_1', f_2', f_3'').

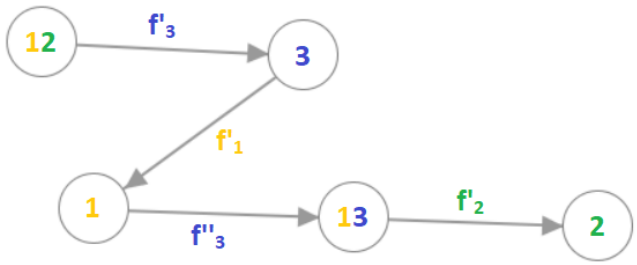

Figure 3.8: Deviation graph for G, Case 3.1c (2)

Once again, this outcome must admit a b.d. for 1, 3, or both. Let that deviation be (f_1'', f_3''') with possibly $f_1'' = f_1'$ or $f_3''' = f_3''$ but not both. Consider the following cases for the resulting outcome, (f_1'', f_2', f_3'''):

- If only Γ_1, or exactly Γ_1 and Γ_2 were satisfied then we would necessarily have $f_3''' = f_3''$ and the joint strategy (f_1'', f_2') would be a punishment for f_3'', contradiction.
- If exactly Γ_1 and Γ_3 were satisfied then (f_1'', f_3''') would be a punishment for f_2', contradiction.
- If only Γ_3, or exactly Γ_2 and Γ_3 were satisfied then we would necessarily have $f_1'' = f_1'$ and the joint strategy (f_2', f_3''') would be a punishment for f_1', contradiction.

Those exhaust all the possible options for *Case 3.1c*, all of which lead to a contradiction.

Since we have a contradiction in all the cases $3.1a$, $3.1b$, $3.1c$, we can reject our assumption, finishing the proof of *Lemma 3.1.*

Continuing the proof of *Claim 3*, we will now prove a stronger lemma about G.

- *Lemma 3.2:*

Let $\{a, b, c\} = \{1, 2, 3\}$ be the set Ag. There cannot be outcomes (f_a, f_b, f_c), (f_a, f_b, f'_c) of G such that all of the following hold:

- (f_a, f_b, f_c) satisfies Γ_a and Γ_b but not Γ_c
- (f_a, f_b, f'_c) satisfies Γ_c only
- f'_c is a beneficial deviation for c from (f_a, f_b, f_c) to (f_a, f_b, f'_c)

- *Proof of Lemma 3.2*:

For contradiction, assume otherwise. WLOG we can again assume that $a = 1$, $b = 2$, and $c = 3$. The negation of the lemma can then be summarised by a deviation graph.

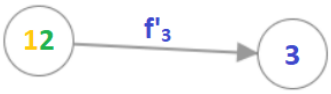

Figure 3.9: Deviation graph for G, Lemma 3.2 (1)

The second outcome, (f_a, f_b, f'_c) must admit a b.d. for 1, 2, or both. However, see that a deviation to an outcome satisfying only Γ_1 or only Γ_2 would contradict *Lemma 3.1.* Hence, the new outcome must have exactly two winners.

Moreover, if the winners in the new outcome were exactly 1 and 2, the deviation would provide a punishment for f'_3. Therefore, WLOG we can assume that (f_a, f_b, f'_c) has a b.d. f'_1 for 1 to an outcome satisfying exactly Γ_1 and Γ_3.

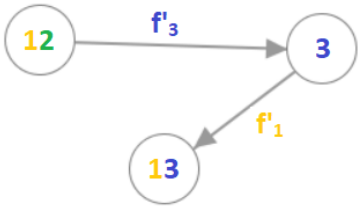

Figure 3.10: Deviation graph for G, Lemma 3.2 (2)

Now, the only loser in (f_1', f_2, f_3'), 2, must have a b.d. f_2'. If however, exactly Γ_1 and Γ_2 were satisfied in (f_1', f_2', f_3'), then (f_1', f_2') would be a punishment for f_3'. If exactly Γ_2 and Γ_3 were satisfied, (f_2', f_3') would provide a punishment for f_1'. Hence, 2 must be the only winner.

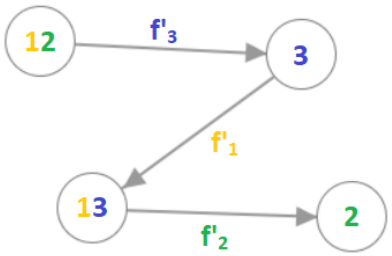

Figure 3.11: Deviation graph for G, Lemma 3.2 (3)

Once again a b.d. from (f_1', f_2', f_3') to an outcome satisfying only Γ_1 or only Γ_3 would contradict *Lemma 3.1.* Therefore, there must be a b.d. by 1, 3, or both, to an outcome with exactly two winners. Now:

- If the winners were 1 and 3, the deviation would provide a punishment for f_2'.
- If the winners were 1 and 2 then the deviation must be f_1'' by 1 only (as 3 loses). But then, the joint strategy (f_1'', f_2') would provide a punishment for the initial f_3'.

- If the winners were 2 and 3 then the deviation must be f_3'' by 3 only. The joint strategy (f_2', f_3'') would be a punishment for f_1'.

This exhausts all the cases, all of which lead to a contradiction, and finishes the proof of *Lemma 3.2*.

Finally, we can strengthten the lemma even further, and prove *Claim 3*: G cannot have an outcome that satisfies exactly two of Γ_1, Γ_2, Γ_3.

For contradiction, assume otherwise. WLOG let (f_1, f_2, f_3) be an outcome satisfying exactly Γ_1 and Γ_2. It must admit a b.d. f_3' for the lone loser, 3. Moreover, by *Lemma 3.2*, he cannot win alone in the new outcome. Therefore, WLOG we can assume that 1 and 3 are winners in (f_1, f_2, f_3').

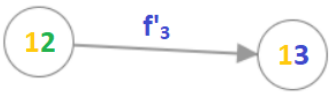

Figure 3.12: Deviation graph for G, Claim 3 (1)

Again, by *Lemma 3.2*, the new outcome must have a b.d. f_2' by 2 to an outcome where exactly two players win. However, if 1 and 2 were winners, then (f_1, f_2') would constitute a punishment for f_3'. Therefore, exactly Γ_2 and Γ_3 must be satisfied in (f_1, f_2', f_3').

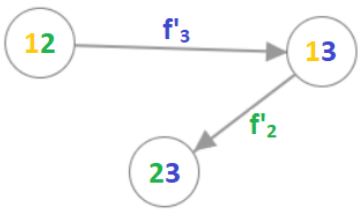

Figure 3.13: Deviation graph for G, Claim 3 (2)

But now, again by *Lemma 3.2*, there must be a b.d. f_1' for 1 to another outcome with exactly two winners. However, if those were 1 and 2, (f_1', f_2') would be a punishment for f_3'. Else, if the winners were 1 and 3 then (f_1', f_3') would form a punishment for f_2'. This exhausts all the cases and finishes the proof of *Claim 3* with a contradiction.

Combining claims 1, 2, and 3 gives us a contradiction for the existence of G, as every concurrent game must have at least one outcome. □

3.3 General case

Usually, most natural problems in game theory and even in general mathematics that consider natural numbers differentiate between at most 3 cases, for $n = 1$, $n = 2$, and $n > 2$. Hence, the existence of a proof for the 3-player case gives an indication that the problem should have the same answer in the general case. In this section, we will show that for core$^+$ such intuition fails. Indeed, we will prove that:

There are concurrent games G for which $core^+(G) = \emptyset$.

3.3.1 Deviation analysis

The biggest hint that led us to believe that the general case might have a different answer than the 3-player case was that the proof for the 2-player case, extended to cover the 3-player case, does not generalise to the n-player case.

To see that, first note the close correspondence between the proof strategy used in Section 3.2 and the idea of deviation graphs. In particular, consider a very important property of (potential) games with no outcome in the core$^+$: *Every game G with $core^+(G) = \emptyset$ must have a deviation graph with a cycle or with an infinite set of*

vertices. This is simply because every node corresponds to an outcome, and every outcome has to admit a beneficial deviation (and hence have an outgoing edge) to another outcome.

In the 3-player case, every attempt to construct such a graph let to a contradiction. Moreover, the contradiction was derived only from the structure of the graph and the definition of core$^+$. Consider, however, the following deviation graph with $|V| = 6$ for a game with $Ag = \{1, 2, 3, 4\}$:

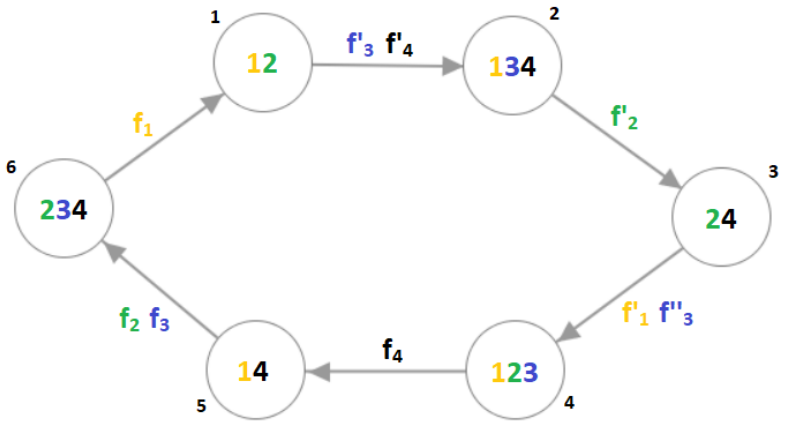

Figure 3.14: Deviation graph for a 4-player game (1)

The function $o : V \to O$ for this graph would be defined as:

$o(1) = (f_1, f_2, f_3, f_4)$

$o(2) = (f_1, f_2, f'_3, f'_4)$

$o(3) = (f_1, f'_2, f'_3, f'_4)$

$o(4) = (f'_1, f'_2, f''_3, f'_4)$

$o(5) = (f'_1, f'_2, f''_3, f_4)$

$o(6) = (f'_1, f_2, f_3, f_4)$

Indeed, it is possible to verify that this graph is not self-contradictory in the way that Figure 3.1 in Section 3.1 was - we could derive the contradiction for the existence

of the game just by looking at the graph. In Figure 3.14 for example, see that in order to form a credible punishment for (f_3', f_4'), we would need to get to an outcome where 1 and 2 win - node 1 or 4. This would however require 3 to not play f_3', and he is not going to punish himself. Similar consistencies exist for other beneficial deviations.

Importantly, the existence of such a graph does not imply the existence of a game with an empty core$^+$-set. It tells us, however, that the proof strategy applied to solve 2 and 3-player cases would not work in the 4-player, nor in the general case.

In fact, it turns out that constructing a counterexample for core$^+$ non-emptiness based on Figure 3.14 is difficult. The reason for that is that the graph only provides an indication of how to construct a small set of outcomes with each having a beneficial deviation. However, in order for the structure to be a legal concurrent game, it has to form an outcome for every set of choices of strategies by players.

For example, in the game that would have Figure 3.14 as a deviation graph, 1, 2, and 4 would choose from at least two strategies, and 3 would choose from at least three. This means that the game would have to have at least 24 outcomes, not just the 6 suggested by the graph, and each of them would have to admit a beneficial deviation.

Moreover, additional outcomes required to complete the game could potentially allow punishments for deviations from currently existing outcomes. The asymetry in roles of the players in Figure 3.14 make it even harder to come up with an example. In practice, it turns out that most "handmade" concurrent games indeed have a core$^+$-stable outcome.

3.3.2 Five-player case

The difficulties presented above led us to believe that it could be easier to solve the 5-player non-emptiness case first. We also start with constructing a deviation graph with a cycle, as that necessarily exists and is significantly easier to find.

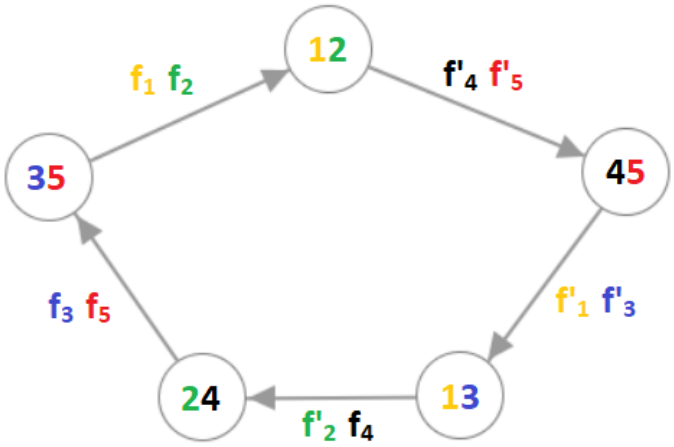

Figure 3.15: Deviation graph for a 5-player game

Notice the improved symmetry and simplicity of this graph over Figure 3.14. Every outcome that forms the cycle has exactly two winners, and every player wins in exactly two outcomes. As a result of that, is suffices that every player chooses between two strategies. The graph satisfies the same properties as Figure 3.14 (it is not self-contradictory) but also presents an idea for a game that is easier to intuitively comprehend.

The CGS for the game that we present is very simple. In the initial state, every player chooses between two strategies, denoted in binary as "0" or "1", and resulting in $2^5 = 32$ total directions - binary strings of length 5, with the i-th bit representing the choice of player i. Depending on that, the play moves to a terminal state (self-looped, without any other outgoing transitions) where some subset of agents win.

Our game, even though it does not actually have the deviation tree presented in Figure 3.15, makes use of an important property that is achieved by that deviation graph. First, given an outcome f call $W(f)$ the *outcome type* of f. See that in a game where all outcomes have exactly two winners, every two outcome types are incomparable (by a subset relation). This means that for every deviation from an outcome of type T, a punishment, in order to be credible, would have to return us to another outcome of type T.

We can make use of this property to exploit the definition of core$^+$ when making a core$^+$-less game in the following way. Suppose that in all outcomes of type $\{1, 2\}$ (the ones with exactly 1 and 2 as winners) player 3 plays "0". If 3 can reasonably deviate (move to an outcome where he wins) from one of those by playing "1" instead, this deviation cannot possibly be punished. This is because returning to an outcome of type $\{1, 2\}$ is impossible without 3 playing "0" again.

In order to construct the game we need to assign each of 32 outcomes to its outcome type. We attempt to organize them in groups of 4 such that every group is assigned to the same outcome type, and has a fixed set of losers deviating to another group. We exploit the idea described above to assure that every attempt for a punishment will be incredible. We use Python for verification of correctness and to search through options when we are stuck.

The intuitive representation of the resulting game is presented below. We display it in 10 lines, each having the form of:
Outcome type: [List of outcomes of this type] - Explanation why every outcome in the list admits a beneficial deviation

Example 3.3.2. *Five-player game with an empty core$^+$-set.*
$\{1, 2\}$: [00000, 00001, 10010, 10011] - For each of those outcomes 3 can beneficially deviate by playing "1", to some outcome of type $\{1, 3\}$. Any attempt to credibly

punish 3 would require him to play “0”.

$\{1,3\}$: [10111, 10110, 00101, 00100, 00111, 00110, 10101, 10100] - For each of those outcomes the coalition of 2 and 4 can deviate to some outcome of type $\{2,4\}$ by playing “1”, “0” or “1”, “1”, respectively. Note that in each of outcomes of type $\{1,3\}$ they play “0”, “0” or “0”, “1”. This means that returning to this outcome type would require their cooperation, and any other punishment is incredible. Moreover, note that some of those deviations only require one of $\{2,4\}$ to change his strategy, even though they form the deviating coalition together. This is consistent with the definition.

$\{1,4\}$: [ ]

$\{1,5\}$: [11111, 11110, 01101, 01100] - The coalition of 3 and 4 can deviate to outcomes of type $\{3,4\}$ by playing “0”, “0” or “0”, “1”, respectively. In each of outcomes of type $\{1,5\}$ they play “1”, “0” or “1”, “1” so any possible punishment would be incredible. Again, here those deviations require only one of the members of the coalition to change his strategy.

$\{2,3\}$: [ ]

$\{2,4\}$: [11101, 11100, 01111, 01110] - The coalition of 3 and 5 can deviate to $\{3,5\}$ by playing “0”, “0” or “0”, “1”. By similar logic, this cannot be credibly punished.

$\{2,5\}$: [ ]

$\{3,4\}$: [01000, 01001, 11010, 11011] - The coalition of 1 and 2 can deviate to $\{1,2\}$ by playing “0”, “0” or “1”, “0”.

$\{3,5\}$: [11000, 11001, 01010, 01011] - The coalition of 1 and 2 can deviate to $\{1,2\}$ by playing “0”, “0” or “1”, “0”.

$\{4,5\}$: [00011, 00010, 10001, 10000] - Here, 3 can deviate to $\{1,3\}$ by playing “1”.

It can be verified that each of the 32 five-bit binary strings representing outcomes is assigned to one of the outcome types, and that no string is repeated. Therefore, this would indeed form a legal concurrent game with no outcome in core$^+$.

More formally, the game can be presented by a CGS graph in Figure 3.16. We have 5 agents, 5 propositional variables (one for each agent), and 8 states with the starting state 0. We define goal sets accordingly with the above example:

$\Gamma_1 = \{(a, *, *, ...) : a \in \{00000, 00001, 10010, 10011, 10111, 10110, 00101, 00100, 00111, 00110, 10101, 10100, 11111, 11110, 01101, 01100\}\}$

$\Gamma_2 = \{(a, *, *, ...) : a \in \{00000, 00001, 10010, 10011, 11101, 11100, 01111, 01110\}\}$

$\Gamma_3 = \{(a, *, *, ...) : a \in \{10111, 10110, 00101, 00100, 00111, 00110, 10101, 10100, 01000, 01001, 11010, 11011, 11000, 11001, 01010, 01011\}\}$

$\Gamma_4 = \{(a, *, *, ...) : a \in \{11101, 11100, 01111, 01110, 01000, 01001, 11010, 11011, 00011, 00010, 10001, 10000\}\}$

$\Gamma_5 = \{(a, *, *, ...) : a \in \{11111, 11110, 01101, 01100, 11000, 11001, 01010, 01011, 00011, 00010, 10001, 10000\}\}$

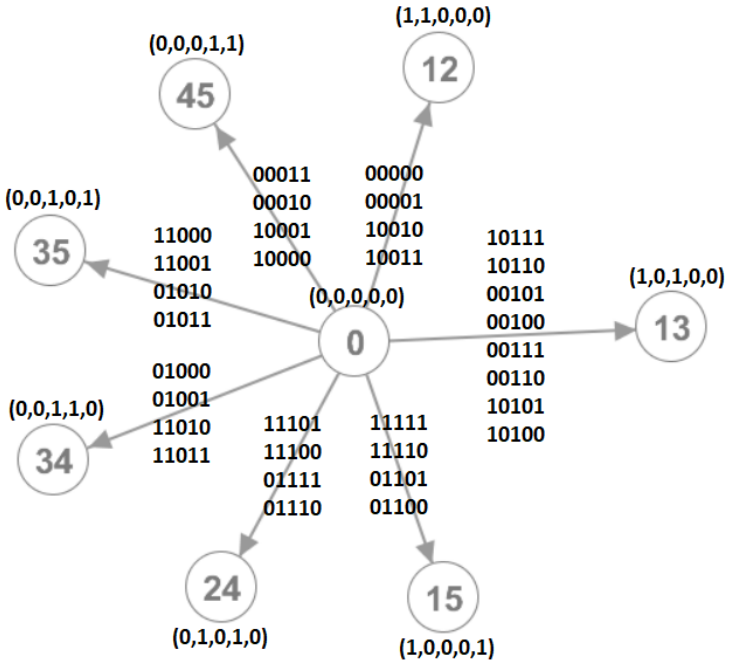

Figure 3.16: CGS for a 5-player game with no core$^+$

It is easy to verify that with such assignment of valuations to states those goals are also run and trace-based. Moreover, as the players can only make a meaningful choice in the initial state, and that is the only state with a valuation $(0, 0, 0, 0, 0)$, the core$^+$-emptiness property holds for all 3 models of strategies.

3.3.3 Four-player case

The last unresolved case for the core$^+$ non-emptiness problem is 4-player games. Combining the deviation analysis from Section 3.3.1 with Example 3.3.2 gives us a good intuition that a counterexample should exist. However, naturally constructing such a game seems difficult.

The idea for the game is very similar to the 5-player case. The players will be able to make exactly one meaningful decision, in the initial state, choosing between two strategies each. This will result in the game having 16 possible outcomes. To each outcome we can assign one of the 16 possible sets of winners. This results in $16^{16} = 2^{64}$ games to consider, way too many for a brute-force search, especially that checking for core$^+$-emptiness is a pretty large constant factor.

We can, however, employ a strategy of a dynamic search. Let O be the set of binary strings of length 4 (possible outcomes). A partial function $f : O \to 2^{Ag}$ is called *consistent* if (with an assumption that for each o, $f(o)$ would be the set of winners in o) for each outcome in the domain of f there is a beneficial deviation to another outcome in the domain of f.

The search algorithm works in the following way. Start with a consistent partial assignment of outcomes to subsets of Ag. At every step, attempt to extend the assignment (add a new pair outcome $\to$ subset of Ag) so that it remains consistent. If with this strategy, we achieve $Dom(f) = O$, return the assignment as an example for a game with no outcome in core$^+$. See that for every partial assignment, there

is at most $16 * 16 = 256$ (choose an outcome and a subset of Ag) possible extensions and the algorithm will make at most 16 steps, so it has a feasible running time.

Obviously, this algorithm does not exhaust the search as it does not check every possible game. Moreover, it might return a false negative, because it could be the case that all assignments with a domain of size n are inconsistent, while there is a consistent assignment of size $n + 1$.

In spite of that, we managed to use the method to find multiple examples of 4-player games without an outcome in core$^+$. See that it is easy to come up with a consistent partial assignment to start the search, given a cyclic deviation graph with each player involved in 2 deviations in the loop. For the example below we used Figure 3.17.

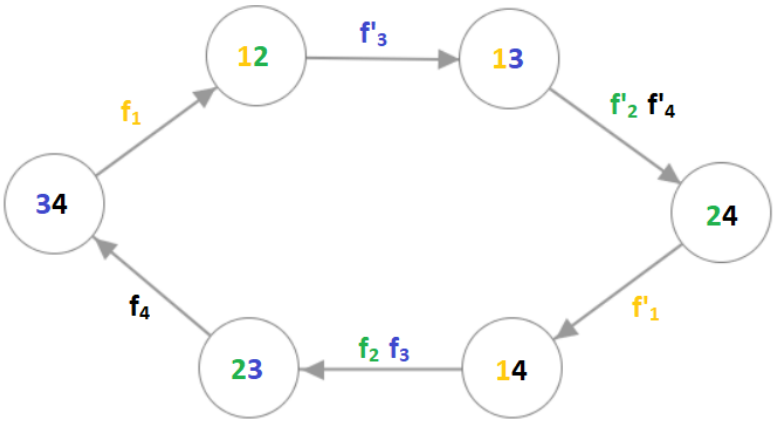

Figure 3.17: Deviation graph for a 4-player game (2)

In order to retrieve the partial assignment, we simply assigned $(f_1, f_2, f_3, f_4) = 0000$ and followed the deviations to get 6 outcomes with the winners given by the labels of the vertices. The algorithm (whose implementation can be found in the appendix) then found the following game, which we also verified manually.

Theorem 3.3. *There is a 4-player game G with $core^+(G) = \emptyset$*

Proof. We give the example below, in the following format:

Outcome: Set of winners - Example of a beneficial deviation

$(0000, *, *, ...) : \{1, 2\}$ - Player 3 can deviate to 0010 by playing "1" instead. In all outcomes where 1 and 2 win, 3 plays 0, hence any punishment would be incredible.

$(0001, *, *, ...) : \{1, 2\}$ - 3 can deviate to 0011.

$(0010, *, *, ...) : \{1, 3\}$ - The coalition of 2 and 4 can deviate to 0111 by both playing "1". In all outcomes where 1 and 3 win, at least one of $\{2, 4\}$ plays "0".

$(0011, *, *, ...) : \{1, 3\}$ - 2 and 4 can deviate to 0111.

$(0100, *, *, ...) : \{1, 2\}$ - 3 can deviate to 0110.

$(0101, *, *, ...) : \{1, 4\}$ - 2 can deviate to 0001.

$(0110, *, *, ...) : \{1, 3\}$ - 2 and 4 can can deviate to 0111.

$(0111, *, *, ...) : \{2, 4\}$ - 1 can deviate to 1111.

$(1000, *, *, ...) : \{3, 4\}$ - 1 can deviate to 0000.

$(1001, *, *, ...) : \{2, 3\}$ - 1 can deviate to 0001.

$(1010, *, *, ...) : \{2, 3\}$ - 1 can deviate to 0010.

$(1011, *, *, ...) : \{2, 3\}$ - 1 can deviate to 0011.

$(1100, *, *, ...) : \{1, 2\}$ - 3 can deviate to 1110.

$(1101, *, *, ...) : \{1, 4\}$ - 2 can deviate to 1001.

$(1110, *, *, ...) : \{2, 3\}$ - 1 can deviate to 0110.

$(1111, *, *, ...) : \{1, 4\}$ - 2 can deviate to 1011.

Once again, we can see that deviations by coalitions do not always require everyone to change their strategy. The CGS for this game is given by Figure 2.1. We have 4 agents, 4 propositional variables and $St = \{0, 12, 13, 14, 23, 24, 34\}$, with the initial state $s_0 = 0$. We define the goal sets as:

$\Gamma_1 = \{(a, *, *, ...) : a \in \{0000, 0001, 0010, 0011, 0100, 0101, 0110, 1100, 1101, 1111\}\}$

$\Gamma_2 = \{(a, *, *, ...) : a \in \{0000, 0001, 0100, 0111, 1001, 1010, 1011, 1100, 1110\}\}$

$\Gamma_3 = \{(a, *, *, ...) : a \in \{0010, 0011, 0110, 1000, 1001, 1010, 1011, 1110\}\}$

$\Gamma_4 = \{(a, *, *, ...) : a \in \{0101, 0111, 1000, 1101, 1111\}\}$

Similarly to Example 3.3.2, these goals are also run and trace-based, and the game has no outcome in core$^+$ for all 3 models of strategies. □

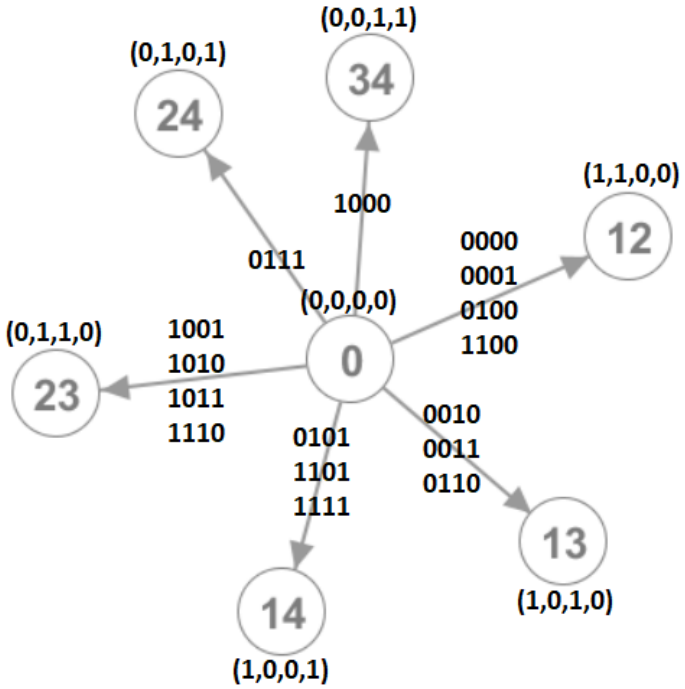

Figure 3.18: CGS for a 4-player game with no core$^+$

Chapter 4

Bisimulation Invariance Problem

The E-core$^+$ problem described in Section 2.6 asks a question whether some outcome in core$^+$ satisfies a given LTL formula. In this chapter, we are interested in checking whether this is a bisimulation invariant property. In other words, similarly to [12] for the case of Nash equilibria, we want to see whether two bisimilar games have "the same" core$^+$-sets, in the sense that the outcomes in those sets satisfy the same temporal logic formulas.

We therefore ask the following question:

Given concurrent games $G = (M, \Gamma_1, ..., \Gamma_n)$ *and* $G' = (M', \Gamma_1, ..., \Gamma_n)$ *where* $M \sim M'$*, does E-core*$^+(G, \phi) \iff$ *E-core*$^+(G', \phi)$ *hold for every LTL formula* ϕ*?*

4.1 Computation and Trace Based Strategies

To begin with, consider the problem for computation and trace-based models of strategies. It was shown in [16] that core$^+$ can be characterised using Strategy Logic (SL) [12, 22]. In particular, given a game $G = (M, \Gamma_1, ..., \Gamma_n)$ and an LTL formula ϕ, there is an SL formula ψ such that E-core$^+(G, \phi)$ if and only if ψ holds.

Moreover, in [12] it has been shown that the same Strategy Logic is invariant under

bisimilarity for both computation and trace-based models of strategies. If a concurrent game in one of those models satisfies a SL formula, the same game on any bisimilar CGS also will. Therefore, we have the following results:

Corollary 4.1.1. *Let G and G' be concurrent games on bisimilar CGSs with the same goals and computation-based model of strategies. Then, E-core$^+(G, \phi) \iff$ E-core$^+(G', \phi)$ for every LTL formula ϕ.*

Proof. Let ϕ be an LTL formula. Let ψ be an SL formula such that ψ holds iff E-core$^+(G, \phi)$. Let ψ' be an SL formula such that ψ' holds iff E-core$^+(G,' \phi)$. By invariance of SL under bisimilarity in the computation-based model of strategies, we have $\psi \iff \psi'$. Therefore, E-core$^+(G, \phi) \iff$ E-core$^+(G', \phi)$. □

Corollary 4.1.2. *Let G and G' be concurrent games on bisimilar CGSs with the same goals and trace-based model of strategies. Then, E-core$^+(G, \phi) \iff$ E-core$^+(G', \phi)$ for every LTL formula ϕ.*

4.2 Computation Based Goals

In this section, we will consider the same problem under the standard, run-based model of strategies but computation-based goals. We will show that, similarly to the case of Nash equilibrium [12], LTL properties of outcomes in core$^+$ are not preserved between bisimilar games.

The intuition for the counterexample is similar to the idea from [12]. Under the run-based model of strategies, the choice that a player makes has to be consistent given a state history. However, in a bisimilar structure, we can "split" a state depending on a computation that led to it, as already shown in Figure 2.3. Intuitively, by doing that we can give a player more power and hence change the core$^+$-set.

Theorem 4.2. *There is a pair of 2-player concurrent games* $G = (M, \Gamma_1, \Gamma_2)$ *and* $G' = (M', \Gamma_1, \Gamma_2)$ *with* $M \sim M'$ *and an LTL formula* ϕ *such that E-core*$^+(G', \phi)$ *but not E-core*$^+(G, \phi)$.

Proof. Consider the CGS M given by Figure 4.1 and M' given by Figure 4.2. Both CGSs have 2 players, and the same set of propositional variables $AP = \{p, q\}$.

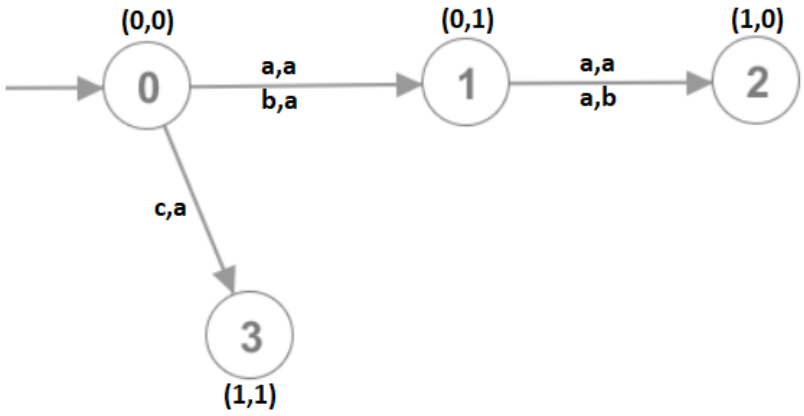

Figure 4.1: CGS M for 2-players bisimulation invariance counterexample

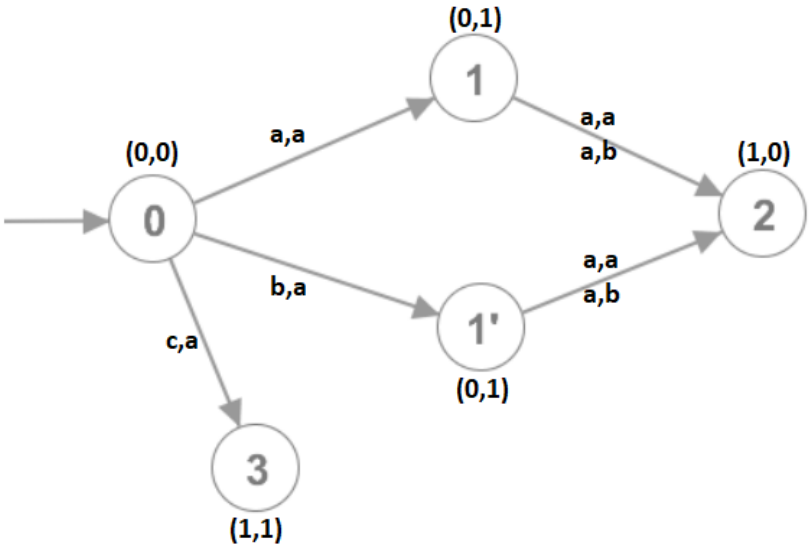

Figure 4.2: CGS M' for 2-players bisimulation invariance counterexample

Sets of states of M and M' are respectively given by $St = \{0, 1, 2, 3\}$ and $St' = \{0, 1, 1', 2, 3\}$, 0 being the initial state of both CGSs. The bisimulation between state 0 of M and state 0 of M' is given by a relation $R : St \times St'$ such that $R = \{(0, 0), (1, 1), (1, 1'), (2, 2), (3, 3)\}$. Hence, we have $M \sim M'$.

Now, let $G = (M, \Gamma_1, \Gamma_2)$ and $G' = (M', \Gamma_1, \Gamma_2)$ be concurrent games with:
$\Gamma_1 = \{(aa, aa, *, *, ...), (ba, ab, *, *, ...)\}$
$\Gamma_2 = \{(ca, *, *, ...)\}$

Consider the LTL formula $\phi = F(p \wedge q)$ expressing "eventually both p and q hold". See that the only (infinite) computation that induces a trace satisfying this formula is $\kappa = (ca, *, *, ...)$ for both M and M'. Moreover, the only outcomes that result in this computation are the ones where player 1 plays "c" in the initial state.

We will first show that E-core$^+(G, \phi)$ fails. See that the only meaningful choices to be made by players are between "a", "b" and "c" in the run (0) for player 1, and between "a" and "b" in the run $(0, 1)$ for player 2. We already know that to have a chance of satisfying ϕ, 1 must play "c" in (0).

Suppose now that 2 plays "a" given the run $(0, 1)$. This outcome is not in core$^+$, because 1 can beneficially deviate and play "a" in (0) instead - the resulting computation would be $(aa, aa, *, ...) \in \Gamma_1$. The potential punishment by 2 changing to "b" in $(0, 1)$ would be incredible, because the initial computation, $(ca, *, ...)$ was in Γ_2, but $(aa, ab, *, ...)$ is not.

Alternatively, suppose that 2 plays "b" in $(0, 1)$. This outcome is also not in core$^+$, because 1 can beneficially deviate by playing "b" in (0) - resulting in the computation $(ba, ab, *, ...) \in \Gamma_1$. Symmetrically, 2 changing to "a" would be an incredible punishment, because he was winning with $(ca, *, ...)$ but he does not win with $(ba, aa, *, ...)$.

Therefore, E-core$^+(G, \phi)$ indeed does not hold.

On the other hand, consider a strategy profile for G' where 1 plays "c" in (0), and 2 plays "b" in $(0,1)$ but "a" in $(0,1')$. The resulting computation is $(ca,*,...) \in \Gamma_2$.

Therefore, the only player who would like to deviate is 1. However, if he decides to play "a" in (0) instead, the resulting computation will be $(aa,ab,*,...) \notin \Gamma_1$. If he decides to play "b", the resulting coputation will be $(ba,aa,*,...) \notin \Gamma_1$. So 1 cannot beneficially deviate. This strategy profile is in core$^+$ and the trace induced by its associated computation satisfies ϕ.

Hence, E-core$^+(G',\phi)$ holds while E-core$^+(G,\phi)$ does not. □

4.3 Run and Trace Based Goals

Importantly, the goals of players in concurrent games in Theorem 4.2 were not trace-based or run-based. We exploited the possibility of defining preferences over computations in order to demonstrate how bisimilarity can add new outcomes into the core$^+$-set. In this section we will show that a similar behaviour is possible in 3-player games for trace-based goals (that are also run-based by definition).

Theorem 4.3. *There is a pair of 3-player concurrent games $G = (M,\Gamma_1,\Gamma_2,\Gamma_3)$ and $G' = (M',\Gamma_1,\Gamma_2,\Gamma_3)$ with trace-based goals and $M \sim M'$, and an LTL formula ϕ such that E-core$^+(G',\phi)$ but not E-core$^+(G,\phi)$.*

Proof. Consider the CGS M given by Figure 4.3 and M' given by Figure 4.4. Both CGSs have 3 players, and the same set of propositional variables $AP = \{p,q\}$.

Sets of states of M and M' are respectively given by $St = \{0,1,2,3,4\}$ and $St' = \{0,1,2,3,4,4'\}$, 0 being the initial state of both. The bisimulation between the initial states is the relation $R : St \times St'$ with $R = \{(0,0),(1,1),(2,2),(3,3),(4,4),(4,4')\}$. Hence, we have $M \sim M'$.

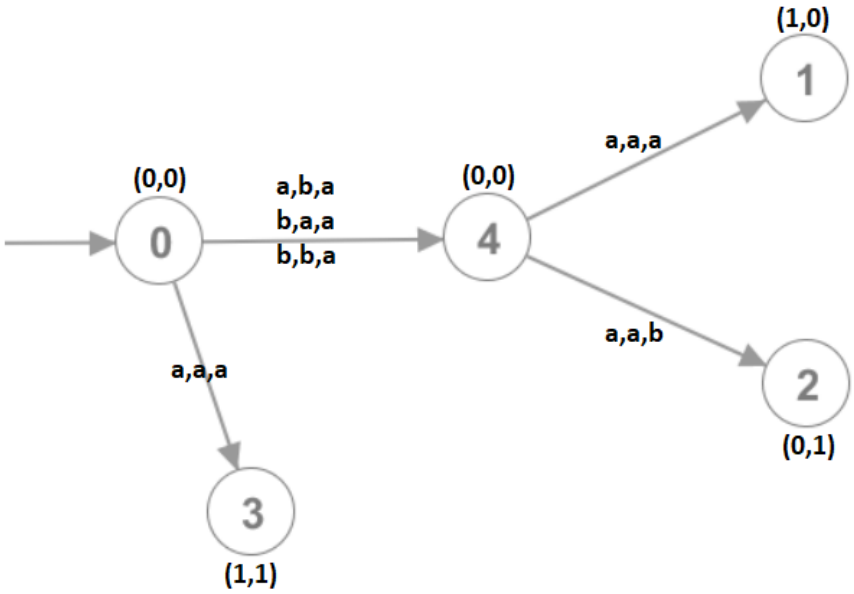

Figure 4.3: CGS M for 3-players bisimulation invariance counterexample

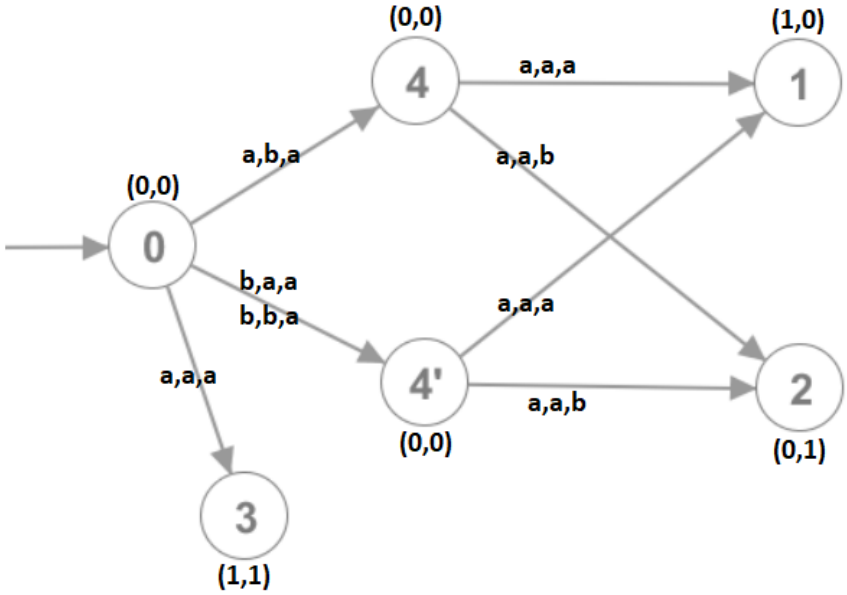

Figure 4.4: CGS M' for 3-players bisimulation invariance counterexample

Now, let $G = (M, \Gamma_1, \Gamma_2, \Gamma_3)$ and $G' = (M', \Gamma_1, \Gamma_2, \Gamma_3)$ be concurrent games with:

$\Gamma_1 = \{(aba, aaa, *, ...), (baa, aaa, *, ...), (bba, aaa, *, ...)\}$

$\Gamma_2 = \{(aba, aab, *, ...), (baa, aab, *, ...), (bba, aab, *, ...)\}$

$\Gamma_3 = \{(aaa, *, *, ...)\}$

It is easy to check that 1 wins exactly in the trace $(0,0)(0,0)(1,0)^\omega$, 2 wins in the trace $(0,0)(0,0)(0,1)^\omega$ and 3 wins in the trace $(0,0)(1,1)^\omega$, therefore these goals are indeed trace-based.

Now, once again consider the LTL formula $\phi = F(p \wedge q)$. See that the only (infinite) computation that induces a trace satisfying this formula is $\kappa = (aaa, *, *, ...)$ for both M and M' (in which case 3 wins). Moreover, the only outcomes that result in this computation are the ones where both 1 and 2 play "a" in the initial state.

We will first show that E-core$^+(G, \phi)$ fails. See that the only meaningful choices to be made by players are between "a", "b" in the run (0) for players 1 and 2, and between "a" and "b" in the run $(0, 4)$ for player 3. We already know that to have a chance of satisfying ϕ, 1 and 2 must both play "a" in (0).

Suppose now that 3 plays "a" given the run $(0, 4)$. This outcome is not in core$^+$, because 1 can beneficially deviate and play "b" in (0) instead - the resulting computation would be $(baa, aaa, *, ...) \in \Gamma_1$. Now, see that regardless of what punishment 2 and 3 decide to apply, they cannot go back to $(aaa, *, *, ...)$ - the only computation where 3 wins again. Hence, any possible punishment would be incredible.

Alternatively, suppose that 3 plays "b" in $(0, 4)$. This outcome is also not in core$^+$, because 2 can beneficially deviate by playing "b" in (0) - resulting in the computation $(aba, aab, *, ...) \in \Gamma_2$. Similarly, 1 and 3 cannot credibly punish this deviation, because they cannot go back to $(aaa, *, *, ...)$.

Therefore, E-core$^+(G, \phi)$ indeed does not hold.

On the other hand, consider a strategy profile for G' where 1 and 2 both play "a" in (0), and 3 plays "b" in $(0,4)$ but "a" in $(0,4')$. The resulting computation is $(aaa, *, ...) \in \Gamma_3$.

Therefore, only 1, 2, or both would like to deviate. Firstly, they certainly cannot deviate together because there is no computation where they both win. Moreover, if 1 decides to deviate by playing "b" in (0), the resulting computation will be $(baa, aab, *, ...) \notin \Gamma_1$. So he cannot beneficially deviate. If 2 decides to play "b" in (0), the resulting coputation will be $(aba, aaa, *, ...) \notin \Gamma_2$. So 2 cannot beneficially deviate either. This strategy profile is in core$^+$ and the trace induced by its associated computation satisfies ϕ.

Hence, E-core$^+(G', \phi)$ holds while E-core$^+(G, \phi)$ does not. □

Since trace-based goals are also run-based, we also immediately have:

Corollary 4.4. *There is a pair of 3-player concurrent games $G = (M, \Gamma_1, \Gamma_2, \Gamma_3)$ and $G' = (M', \Gamma_1, \Gamma_2, \Gamma_3)$ with run-based goals and $M \sim M'$, and an LTL formula ϕ such that E-core$^+(G', \phi)$ but not E-core$^+(G, \phi)$.*

Those results solve the general case for bisimulation invariance of core$^+$, but there might still be a special case for 2-player games with run or trace-based goals. In [13] it was shown that in fact, Nash equilibria are preserved under bisimilarity in 2-player games when the preferences are run-based. For the core$^+$, we unfortunately have to leave it as an open question.

Chapter 5

Non-Emptiness Invariance Under Bisimilarity

The last model theoretic question we might want to ask about core$^+$ and bisimilarity is whether at least the existence of any core$^+$-stable outcome is invariant between bisimilar game structures. In the case of Nash equilibria this turned out to have the same answers for specific n-player cases as the question asked in Chapter 4 [12, 13]. Here however, given the results from Chapters 3 and 4 we already know that this is not the case. We ask whether:

Given concurrent games $G = (M, \Gamma_1, ..., \Gamma_n)$ *and* $G' = (M, \Gamma_1, ..., \Gamma_n)$ *where* $M \sim M'$*, does* $core^+(G) \neq \emptyset \iff core^+(G') \neq \emptyset$ *hold?*

5.1 Computation and Trace Based Strategies

Given the result from Section 4.1 of Chapter 4, the answer is trivial for the cases of computation and trace-based strategies.

Corollary 5.1.1. *Let* G *and* G' *be concurrent games on bisimilar CGSs with the same goals and computation-based model of strategies. Then,* $core^+(G) \neq \emptyset \iff core^+(G') \neq \emptyset$.

Proof. Every outcome satisfies the LTL formula $\phi = True$. Therefore, given any con-

current game G, we have $core^+(G) \neq \emptyset$ if and only if E-core$^+(G, True)$. By corollary 4.1.1, plugging in $\phi = True$ we have E-core$^+(G, True) \iff$ E-core$^+(G', True)$. We conclude that $core^+(G) \neq \emptyset \iff core^+(G') \neq \emptyset$. □

Corollary 5.1.2. *Let G and G' be concurrent games on bisimilar CGSs with the same goals and trace-based model of strategies. Then, $core^+(G) \neq \emptyset \iff core^+(G') \neq \emptyset$.*

5.2 Run Based Strategies

In the case of run-based strategies, the solution is significantly less clear. We know for certain that at least in the 3-player case, the answer is different to the problem from Chapter 4. This is because by Theorem 3.2, core$^+$ is always non-empty in 3-player games, and hence its non-emptiness is trivially preserved under bisimilarity. We have:

Corollary 5.2. *For any pair of 2 or 3-player games $G = (M, \Gamma_1, ..., \Gamma_n)$ and $G' = (M, \Gamma_1, ..., \Gamma_n)$ where $M \sim M'$, we have $core^+(G) \neq \emptyset \iff core^+(G') \neq \emptyset$.*

However, by Theorem 4.3, LTL properties of those sets are not preserved.

On the other hand, we know by Theorem 3.3 that core$^+$ can be empty in the general case. Moreover, from examples in Theorem 4.2 and Theorem 4.3 we can see that bisimilarity in cases of paths with at least 3 distinct states can add new outcomes into the core$^+$-set.

Therefore, the intuition would suggest that we could construct a more complex instance of a 4-player game with empty core$^+$ and then use the strategy from Chapter 4 of "splitting" states. This way we would create a bisimilar CGS that would add an outcome to the core$^+$-set, providing a counterexample to the claim.

However, counterexamples presented in Chapter 3 are already quite complicated and were only found with the help of a computer. It seems even more difficult to build a core$^+$-less game with the required property - a possibility to split states in bisimulation in order to push outcomes into core$^+$, like presented in Chapter 4. In such a game, players would potentially need to choose between more than 2 strategies, which would make search and verification extremely costly.

Based on experience, concurrent games with no outcomes in core$^+$ are quite rare and finding them requires some clever engineering. For those reasons, we for now have to leave the general case of this problem as an open question.

Chapter 6

Other Cooperative Solution Concepts

The main motivation for using core$^+$ instead of the better established α-core concept is to handle the credibility of punishments [16]. Instead of allowing agents to execute any action in order to punish the deviators, we try and make sure that this behaviour would also be rational. To do that, we require the outcome obtained as a result of the punishment to at least maintain the winners from the initial outcome.

Our main criticism of this definition, developed over the course of the study on core$^+$ is that this requirement is in fact too strong. In real world scenarios, people often act against their interests to achieve revenge. In fact, if one is indifferent between two outcomes but has an opportunity to punish an opponent who stops them from achieving their goal, it could be considered rational to do so.

Secondly, core$^+$ fails to take into account that not all of the winners in the stable outcome have to be involved in the coalition attempting to maintain it. Indeed, it requires all of the initial winners to achieve their goals again when any deviation is being punished. Considering that they might "not have done anything" to be winners in the first place, this seems too strong of a condition.

Finally, model theoretic properties of core$^+$, albeit surprising, turn out to be negative. Contrary to the α-core, core$^+$ may be empty in multi-player games, and is not

preserved by bisimilarity under the standard model of strategies.

Keeping these remarks in mind, in this chapter we suggest alternative variations of the core. We aim to achieve a similar goal of making punishments more credible but with a more natural solution concept that could potentially have better properties.

6.1 Core*

In the definition of core$^+$, three outcomes are considered in total. The initial outcome, the goal of the deviators (intermediate outcome) and the outcome after the punishment is applied (final outcome). The argument is that in order for the punishment to be credible, all winners from the initial outcome have to be maintained in the final outcome.

However, as we mentioned earlier, in a way this attempt tries to make punishments "too rational", ignoring the essence of what a punishment is. The main goal of a punishment in human sense, is still to serve justice to the perpetrator. The punishment would be credible if by doing that we are not harming our cause. Therefore, instead of the initial outcome, to see whether we are not harming ourselves by applying the punishment we should be looking at the intermediate outcome.

We say that f'_C, a joint strategy for some $C \subseteq Ag$ is *beneficial deviation* from an outcome f if all of the following hold:

- $C \subseteq L(f)$
- $C \subseteq W(f_{-C}, f'_C)$
- for every subset $A \subseteq Ag \setminus C$ and every joint (punishment) strategy f'_A for A:
  - either $C \subseteq W(f_{-C \cup A}, f'_C, f'_A)$
  - or there is some $a \in A$ such that $a \in W(f_{-C}, f'_C)$ and $a \in L(f_{-C \cup A}, f'_C, f'_A)$

Intuitively, this time we say that a punishment by a subset of agents is credible if as a result of applying it none of them go from winning to losing.

We say that an outcome f is *in the core** of a game G if there does not exist a beneficial deviation from f as above.

Even though this definition does not handle the exact example given in [16] as criticism of the core, it makes punishments more credible in a different way. Consider a 2-player game on a CGS given by 6.1 with the initial state 0.

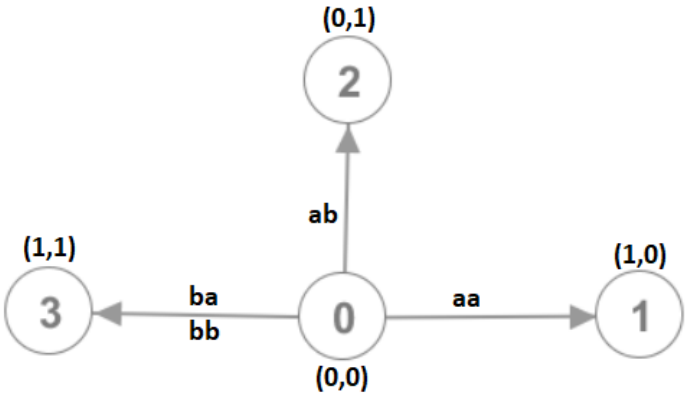

Figure 6.1: A game with a (very) incredible punishment

Let the goals of the agents be:

$\Gamma_1 = \{(aa, *, ...), (ab, *, ...)\}$

$\Gamma_2 = \{(ab, *, ...)\}$

Here, 1 wins if play reaches either state 1 or state 2, and 2 wins if play reaches state 2. See that even now, the outcome where both 1 and 2 play "a" is in the α-core, because if 2 tried to deviate and play "b", 1 could punish him by playing "b" as well. This punishment is however, even more incredible than in the example in [16], as its execution directly harms 1. With our definition of core* we can at least discard such outcomes as unstable.

Moreover, we can prove the following claim about core*:

Theorem 6.1. *core*$^*(G) \neq \emptyset$ *for every concurrent game* G

Proof. For contradiction, suppose otherwise. Fix an outcome f_0 of G. Then, there must be an infinite sequence of outcomes $(f_0, f_1, ...)$, an infinite sequence of subsets of Ag, $(C_1, C_2, ...)$, and an infinite sequence of joint strategies for those subsets $(f'_{C_1}, f'_{C_2}, ...)$ such that for each i, f'_{C_i} is a beneficial deviation (b.d.) from f_{i-1} to f_i.

Given two joint strategies f_A, f_B for sets A and B, respectively, we will write $f_A \circ f_B$ for their composition. By that we mean a joint strategy for $A \cup B$ where each player in B plays his strategy from f_B and each player in A, but not in B plays his strategy from f_A.

We will now use strong induction to prove that for each natural number n, we have: $\bigcup_{i=1}^{n} C_n \subseteq W(f_n)$.

1. Base case, $n = 1$.

As f'_{C_1} is a b.d. to f_1, we must have $C_1 \subseteq W(f_1)$.

2. Inductive step, $n \geq 2$.

We assume that for each $j \geq 1$; $j \leq n-1$, we have $\bigcup_{i=1}^{j} C_j \subseteq W(f_j)$ - outer IH.

Now, by strong induction on the set $K = \{n, n-1, ..., 1\}$ we will show that for each $k \in K$, $C_k \subseteq W(f_n)$.

2.1. Base case, $k = n$.

As f'_{C_n} is a b.d. to f_n, we must have $C_n \subseteq W(f_n)$.

2.2. Inductive step, $1 \leq k \leq n-1$.

We assume that for each $i > k$; $i \leq n$, we have $C_i \subseteq W(f_n)$ - inner IH.

So $\bigcup_{i=k+1}^{n} C_i \subseteq W(f_n)$.

See that by outer IH, all of C_k win in all of $f_k, f_{k+1}, ..., f_{n-1}$. Hence none of them can be a member of any of the b.d.s $f'_{C_{k+1}}, f'_{C_{k+2}}, ..., f'_{C_n}$. Therefore, we can consider their composition, a joint strategy $f'_C = f'_{C_{k+1}} \circ f'_{C_{k+2}} \circ ... \circ f'_{C_n}$ for $C = \bigcup_{i=k+1}^{n} C_i$ as a punishment for f'_{C_k}. See that applying this punishment would take us to the outcome f_n.

Since by assumption f'_{C_k} is beneficial we must have $C_k \subseteq W(f_n)$, or else there would be some $a \in C$ with $a \in L(f_n)$ which would contradict the inner IH.

This finishes step 2.2 and the inner induction. For each $k \in \{n, n-1, ..., 1\}$ we have $C_k \subseteq W(f_n)$ so we can conclude that $\bigcup_{i=1}^{n} C_i \subseteq W(f_n)$. This in turn finishes step 2 and the outer induction.

But now, we know that for each natural number n, we have: $\bigcup_{i=1}^{n} C_n \subseteq W(f_n)$. Moreover, for each natural number $n \geq 1$, $C_n \subseteq L(f_{n-1})$, and each of the sets C_i is non-empty. Therefore, the infinite sequence of sets $(C_1, C_1 \cup C_2, C_1 \cup C_2 \cup C_3, ...)$ is strictly increasing in size, which is a contradiction as Ag is finite. □

6.2 Core'

Our other suggestion addresses the second concern mentioned in the introduction to this chapter. When a deviation is considered, not every winner in the initial outcome has to be a member of the opposing coalition. This definition aims to achieve a better solution concept by weakening the credibility requirements of core$^+$. Specifically, when a punishment is declared, we only require players who actually execute it to go back to winning, if they indeed were winners in the initial outcome.

We say that f'_C, a joint strategy for some $C \subseteq Ag$ is a *beneficial deviation* from an outcome f if all of the following hold:

- $C \subseteq L(f)$
- $C \subseteq W(f_{-C}, f'_C)$
- for every subset $A \subseteq Ag \setminus C$ and every joint (punishment) strategy f'_A for A: $A \subseteq W(f_{-C \cup A}, f'_C, f'_A) \cup L(f) \Longrightarrow C \subseteq W(f_{-C \cup A}, f'_C, f'_A)$

Intuitively, if a player who won in the initial outcome wants to participate in a punishment, he also has to win in the final outcome. This is a strictly weaker requirement than in the case of core$^+$, where all initial winners had to remain winners, not just the ones who punish.

We say that an outcome f is *in the core'* of a game G if there does not exist a beneficial deviation from f as defined above.

To see an example of unintuitive behaviour of core$^+$ that core' does not exhibit, consider a 3-player game played on a CGS presented in Figure 6.2 with the initial state 0.

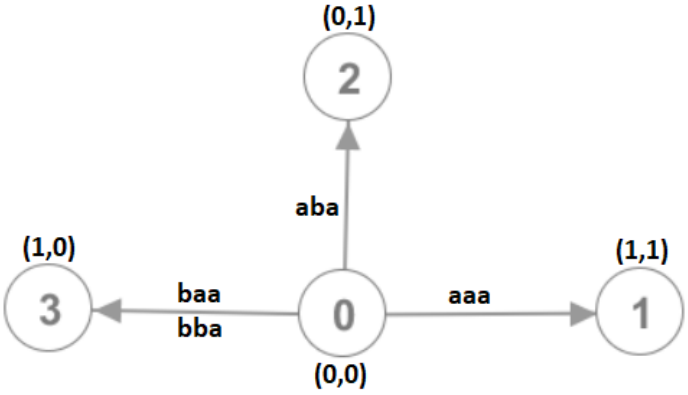

Figure 6.2: A game with a credible punishment

Let the goals of players be:

$\Gamma_1 = \{(aaa, *, ...), (baa, *, ...), (bba, *, ...)\}$

$\Gamma_2 = \{(aba, *, ...)\}$

$\Gamma_3 = \{(aaa, *, ...)\}$

So that 1 wins if the play reaches state 1 or 3, 2 wins if it reaches state 2, and 3 wins only if it reaches state 1. See that in this game, player 3 does not even have a meaningful decision to make - he only has one available strategy. However, his presence in the game, and the fact that he wins in state 1, takes the outcome where all players play "a" out of core$^+$.

If player 3 was not in the game, or by that note did not win in state 1, the deviation by 2 to "b" would not be beneficial in the context of core$^+$, as 1 could credibly punish it by also playing "b". Here, even though 3 is essentially a dummy player, he suddenly makes that punishment incredible.

On the other hand, as core' only requires players that actually execute the punishment to go back to winning, this deviation would not be beneficial. Hence, as our intuition suggests the outcome where all 3 players play "a" is in the core'.

Chapter 7

Conclusions

7.1 Summary of Results

Overall, we managed to prove a number of model theoretic properties of core$^+$. We present the results in 3 tables, each corresponding to a chapter of the thesis and the question discussed in that chapter. The nine entries in each table represent the 9 considered combinations of models for strategies and goals. "Y" stands for a positive answer, "N" for negative, and "?" means an open question.

Core$^+$ non-emptiness			
Goals			
Strategies	Comp-based	Run-based	Trace-based
Comp-based	N[12]	N[12]	N[12]
Run-based	N[12]	N[12]	N[12]
Trace-based	N[12]	N[12]	N[12]

Figure 7.1: Results for core$^+$ non-emptiness

[1] *Based on a 4-player counterexample (Theorem 3.3)*

[2] *Yes for 2-player (Theorem 3.1) and 3-player (Theorem 3.2) games*

Core$^+$ LTL properties preserved under bisimilarity			
Goals			
Strategies	Comp-based	Run-based	Trace-based
Comp-based	Y[1]	Y[1]	Y[1]
Run-based	N[3]	N[5]	N[4]
Trace-based	Y[2]	Y[2]	Y[2]

Figure 7.2: Results for core$^+$ bisimulation invariance

[1] *From results on SL [12, 16] (Corollary 4.1.1)*

[2] *From results on SL [12, 16] (Corollary 4.1.2)*

[3] *Based on a 2-player counterexample (Theorem 4.2)*

[4] *Based on a 3-player counterexample (Theorem 4.3)*

[5] *Based on a 3-player counterexample (Corollary 4.4)*

Core$^+$ non-emptiness preserved under bisimilarity			
Goals			
Strategies	Comp-based	Run-based	Trace-based
Comp-based	Y[1]	Y[1]	Y[1]
Run-based	?[3]	?[3]	?[3]
Trace-based	Y[2]	Y[2]	Y[2]

Figure 7.3: Results for core$^+$ non-emptiness bisimulation invariance

[1] *From above results (Corollary 5.1.1)*

[2] *From above results (Corollary 5.1.2)*

[3] *Yes for 2-player and 3-player games (Corollary 5.2)*

Moreover, by Theorem 6.1 we have the following model theoretic result for the newly introduced concept, core*:

Core* non-emptiness			
Goals			
Strategies	Comp-based	Run-based	Trace-based
Comp-based	Y	Y	Y
Run-based	Y	Y	Y
Trace-based	Y	Y	Y

Figure 7.4: Results for core* non-emptiness

7.2 Open Questions

In respect to the most important model theoretic properties of core$^+$, this paper leaves two questions unanswered. First one is the 2-player potential special case for invariance of core$^+$ under bisimilarity. Precisely, we have not decided whether:

Given 2-player concurrent games $G = (M, \Gamma_1, \Gamma_2)$ *and* $G' = (M', \Gamma_1, \Gamma_2)$ *with run-based strategies, trace-based or run-based goals, and* $M \sim M'$*, does E-core*$^+(G, \phi) \iff$ *E-core*$^+(G', \phi)$ *hold for every LTL formula* ϕ*?*

A similar special case for Nash equilibria is considered in [13], and contrary to the general case, the answer indeed turns out to be positive. The proof is based on the idea of run-invariant equilibria, but also uses the fact that NE in 2-player games can be characterised in terms of winning strategies. It is unclear how to extend that proof to cover the case of core$^+$, but it gives us an intuition that this question might also have a positive answer.

The other open question concerns the preservation of existence of any core$^+$-stable

outcome under bisimulation. Chapter 5 explains how, unlike in the NE case, here it is actually an interesting question. We are interested whether:

Given concurrent games $G = (M, \Gamma_1, ..., \Gamma_n)$ *and* $G' = (M', \Gamma_1, ..., \Gamma_n)$ *with more than 3 players and where* $M \sim M'$*, does* $core^+(G) \neq \emptyset \iff core^+(G') \neq \emptyset$ *hold?*

In this case, the intuition given by results from Chapters 3 and 4 indicates that the answer is likely to be negative. The construction of a counterexample, however, seems highly non-trivial and could potentially require a significant amount of computational power for searching. This is another theoretical problem that could be considered for future research on core$^+$.

7.3 Different Solution Concepts

In Chapter 6 we suggested two alternatives to core$^+$, each addressing different issues with that solution concept. Core* attempts to capture the essence of a punishment and uses an arguably more natural understanding of when it should be considered credible or rational. On the other hand, core' looks at an unjustified assumption that core$^+$ makes about members of the punishing coalitions, and weakens the concept to get rid of some instances of undesired behaviour.

Both options are, in our opinion, a good middle ground between what α-core offers and how core$^+$ handles its inherent irrationality. Even though we conjecture core* to be incomparable in strength to core$^+$, as it looks at winners in the intermediate outcome instead of the initial outcome, we know that it is weaker in the sense that it is always non-empty. On the other hand, core' allows for more punishment strategies than core$^+$, and so is strictly weaker.

Both of the solution concepts suggested by us seem to impose weaker restrictions than core$^+$, which makes them likely to have better model theoretic properties also

in relation to bisimilarity. We propose a deeper analysis of those concepts as future work on cooperative concurrent games.

7.4 Related Work

The questions adressed in this paper are similar to earlier work on Nash equilibria [12] and the core [16] in concurrent games. The former studied model theoretic properties of the Nash equilibrium, and showed that neither its existence, nor temporal logic properties are preserved between bisimilar games, at least not in the most commonly used model of strategies. On the other hand, the latter provided better results for the case of the main cooperative solution concept - α-core from game theory literature. However, it also criticised α-core for allowing incredible strategies in coalition formation and suggested an alternative - core$^+$ which was studied in detail in this thesis.

The two variations of the core introduced in [16] for cooperative games only constitute a small portion of research on coalition formation in game theory [10, 24, 27]. While α-core assumes the worst case punishment for a set of deviators, β-core only requires them to benefit in some counter-coalition scenario. On the other hand, γ-core [5, 6] only allows other players to punish deviators individually (without forming a coalition). Those suggestions are all worth consideration when trying to establish a cooperative solution concept in concurrent games.

In rational verification we attempt to check what temporal logic properties are satisfied by a system where all agents behave rationally [11, 15, 18]. For the most part, such assumption followed non-cooperative game theory and defined rational behaviour through the concept of Nash equilibria [14, 25]. Even though they are the best established solution, NE are known to not always be satisfactory when cooperation is allowed [20, 23]. In [16], complexity of verifying logical properties for stable

outcomes was studied for cooperative concepts of α-core and core$^+$.

Concurrent games are not the only framework for analysis of multi-agent systems. Boolean game structures [15] are considered to be a well behaved class of games in the context of solution concepts and theoretical properties. Indeed, in [12] it was shown that in Boolean game structures Nash equilibria are invariant under bisimilarity for all models of strategies and goals. It could be worthwhile to also analyse the behaviour of cooperative solution concepts in Boolean games, in hope of achieving better model theoretic properties for more complex constructions such as core$^+$.

List of Figures

2.1 Example of a CGS with 2 players 8

2.2 A simple CGS with 1 player 9

2.3 A CGS with 1 player bisimilar to Figure 2.2 9

3.1 Deviation graph for a 2-player game with empty core$^+$ 16

3.2 Deviation graph for G, Claim 2 (1) 18

3.3 Deviation graph for G, Claim 2 (2) 18

3.4 Deviation graph for G, Lemma 3.1 20

3.5 Deviation graph for G, Case 3.1a 20

3.6 Deviation graph for G, Case 3.1b 21

3.7 Deviation graph for G, Case 3.1c (1) 21

3.8 Deviation graph for G, Case 3.1c (2) 22

3.9 Deviation graph for G, Lemma 3.2 (1) 23

3.10 Deviation graph for G, Lemma 3.2 (2) 24

3.11 Deviation graph for G, Lemma 3.2 (3) 24

3.12 Deviation graph for G, Claim 3 (1) 25

3.13 Deviation graph for G, Claim 3 (2) 25

3.14 Deviation graph for a 4-player game (1) 27

3.15 Deviation graph for a 5-player game 29

3.16 CGS for a 5-player game with no core$^+$ 32

3.17 Deviation graph for a 4-player game (2) 34

3.18 CGS for a 4-player game with no core$^+$ 36

4.1 CGS M for 2-players bisimulation invariance counterexample 39

4.2 CGS M' for 2-players bisimulation invariance counterexample 39

4.3 CGS M for 3-players bisimulation invariance counterexample 42

4.4 CGS M' for 3-players bisimulation invariance counterexample 42

6.1 A game with a (very) incredible punishment 50

6.2 A game with a credible punishment 53

7.1 Results for core$^+$ non-emptiness 55

7.2 Results for core$^+$ bisimulation invariance 56

7.3 Results for core$^+$ non-emptiness bisimulation invariance 56

7.4 Results for core* non-emptiness 57

Bibliography

[1] Rajeev Alur, Thomas A. Henzinger, Orna Kupferman. *Alternating-Time Temporal Logic.* Journal of the ACM, Vol. 49, No. 5, 672-713. 2002.

[2] Christel Baier, Joost-Pieter Katoen. *Principles of Model Checking.* MIT Press, 2008.

[3] Patricia Bouyer, Romain Brenguier, Nicolas Markey, Michael Ummels. *Nash Equilibria in Concurrent Games with Büchi Objectives.* IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'11), Mumbai, India, pages 375–386, 2011.

[4] Georgios Chalkiadakis, Edith Elkind, Michael Wooldridge. *Computational Aspects of Cooperative Game Theory.* Morgan & Claypool, October 2011.

[5] Parkash Chander. *Cores of games with positive externalities.* CORE Discussion Papers 2010/04. 2010.

[6] Parkash Chander. *The gamma-core and coalition formation.* International Journal of Game Theory, Vol. 35, 4, 539-556. 2007.

[7] Edmund M. Clarke, Orna Grumberg, Doron A. Peled. *Model Checking.* MIT Press, 2002.

[8] Stéphane Demri, Valentin Goranko, Martin Lange. *Temporal Logics in Computer Science: Finite-state Systems.* Cambridge Tracts in Theoretical Computer Science, Vol. 58. Cambridge University Press, 2016.

[9] E. Allen Emerson. *Temporal and Modal Logic.* Handbook of Theoretical Computer Science, Vol. B. MIT Press, 995-1072. 1990.

[10] Michael Finus, Bianca Rundshagen. *A Non-cooperative Foundation of Core-Stability in Positive Externality NTU-Coalition Games.* Nota Di Lavoro 31.2003. Economics Energy Environment, 2003.

[11] Dana Fisman, Orna Kupferman, Yoad Lustig. *Rational Synthesis.* TACAS (Lecture Notes in Computer Science), Vol. 6015. Springer, 190-204. 2010.

[12] Julian Gutierrez, Paul Harrenstein, Giuseppe Perelli, Michael Wooldridge. *Nash Equilibrium and Bisimulation Invariance.* CONCUR (LIPIcs), Vol. 85. Schloss Dagstuhl, 17:1-17:16, 2017.

[13] Julian Gutierrez, Paul Harrenstein, Giuseppe Perelli, Michael Wooldridge. *Nash Equilibrium and Bisimulation Invariance.* Logical Methods in Computer Science, to appear, 2019. Available at `https://arxiv.org/abs/1808.09213`

[14] Julian Gutierrez, Paul Harrenstein, Michael Wooldridge. *From model checking to equilibrium checking: Reactive modules for rational verification.* Artificial Intelligence 248, 123-157. 2017.

[15] Julian Gutierrez, Paul Harrenstein, Michael Wooldridge. *Iterated Boolean Games.* Information and Computation 242, 53-79. 2015.

[16] Julian Gutierrez, Sarit Kraus, Michael Wooldridge. *Cooperative Concurrent Games.* Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, May 13–17, 2019, IFAAMAS.

[17] Matthew Hennessy, Robin Milner. *Algebraic Laws for Nondeterminism and Concurrency.* Journal of the ACM, 32(1):137–161, 1985.

[18] Orna Kupferman, Giuseppe Perelli, Moshe Y. Vardi. *Synthesis with Rational Environments.* Annals of Mathematics and Artificial Intelligence, Vol. 78, 1, 3-20. 2016.

[19] Alessio Lomuscio, Hongyang Qu, Franco Raimondi. *MCMAS: an open-source model checker for the verification of multi-agent systems.* STTT, Vol. 19, 1, 9-30. 2017.

[20] Michael Maschler, Eilon Solan, Shmuel Zamir. *Game Theory.* Cambridge University Press, 2013.

[21] Robin Milner. *Communication and Concurrency.* Prentice Hall, 1989.

[22] Fabio Mogavero, Aniello Murano, Giuseppe Perelli, Moshe Y. Vardi. *Reasoning About Strategies: On the Model-Checking Problem.* ACM Transactions on Computational Logic, 15(4):1–47, 2014.

[23] Martin J. Osborne, Ariel Rubinstein. *A Course in Game Theory.* MIT Press, 1994.

[24] Metin Uyanik. *On the nonemptiness of the α-core of discontinuous games: Transferable and nontransferable utilities.* Journal of Economic Theory, Vol. 158, 213-231. 2015.

[25] Michael Wooldridge, Julian Gutierrez, Paul Harrenstein, Enrico Marchioni, Giuseppe Perelli, Alexis Toumi. *Rational Verification: From Model Checking to Equilibrium Checking.* Proc. of AAAI. AAAI Press, 4184-4191. 2016.

[26] Michael Wooldridge. *An Introduction to Multiagent systems - Second Edition.* John Wiley & Sons, May 2009.

[27] Sang-Seung Yi. *Stable Coalition Structures with Externalities.* Games and Economic Behavior, Vol. 20, 2, 201-237. 1997.

Appendix

search4p.py - the search algorithm for a 4-player game with no outcome in core$^+$

```
import copy
from itertools import combinations

B = ('0','1')

# sets of winners to be considered for outcomes
POS_WINNERS = [{0,1}, {0,2}, {0,3},
               {1,2}, {1,3},
               {2,3},
               {0,1,2}, {0,1,3}, {0,2,3}, {1,2,3}]

# set Ag
PLAYERS = {0,1,2,3}

# initial (consistent) partial assignment
g1 = {'0000': {0,1},
      '0010': {0,2},
      '0111': {1,3},
      '1111': {0,3},
      '1001': {1,2},
      '1000': {2,3}}

# set of possible outcomes
OUTCOMES = {'0000', '0001', '0010', '0011',
            '0100', '0101', '0110', '0111',
            '1000', '1001', '1010', '1011',
            '1100', '1101', '1110', '1111'}

def flip(b):
    if b == '0': return '1'
    else: return '0'

def flipbits(out, bits):
    out2 = ''
    for i in range(4):
        if i in bits:
            out2 += flip(out[i])
        else:
            out2 +=out[i]
    return out2

def nocore(game):  # verify that a partial assignment is consistent
    return all(
        hasBD(game, out) for out in game)

def subsets(players):  # get all non-empty subsets of a subset of Ag
    subs = []
    for sub in combinations(players, 1):
        subs.append(set(sub))
    for sub in combinations(players, 2):
        subs.append(set(sub))
    for sub in combinations(players, 3):
        subs.append(set(sub))
    return subs

def deviations(out, players):   # get the list of possible deviations for
a set
    dev = []                    # of players from a given outcome
    for sub in subsets(players):
        dev.append(flipbits(out, sub))
    return dev

def hasBD(game, out):       # check if a particular outcome admits a b.d.
    return any(canBD(game, out, coal) for coal in subsets(PLAYERS -
```

```
        game[out]))
64
65  def canBD(game, out, coal):  # check if a particular coalition can b.d.
    from an outcome
66      return any(out2 in game
67                 and not (coal - game[out2])   # coalition wins in out2
68                 and isBeneficial(game, out, out2, coal) for out2 in
                   deviations(out, coal))
69
70  def isBeneficial(game, out, out2, coal):
71      return all(out3 not in game               # the punishment not in the
        game
72                 or (game[out] - game[out3])    # or out3 drops a winner
                   from out
73                 or not (coal - game[out3])     # or coal remains winners
                   in out3
74                 for out3 in deviations(out2, PLAYERS - coal))
75
76  def addoutcome(game, out, winners):
77      g2 = copy.deepcopy(game)
78      g2[out] = winners
79      return g2
80
81  def recsearch(game):    # recursively attempt to add outcomes to the
    partial assignment
82      if len(game) == 16:
83          return game
84      else:
85          for out in (OUTCOMES - set(game)):
86              for win in POS_WINNERS:
87                  newG = addoutcome(game, out, win)
88                  if nocore(newG):
89                      res = recsearch(addoutcome(game, out, win))
90                      if res is not None:
91                          return res
92          return None
93
94  def printgame(game):
95      for out in sorted(game):
96          print(out + ': ' + str(game[out]))
97
98  G = recsearch(g1)
99  printgame(G)
100 print(nocore(G))
101
```